Informix Product Family
Informix
Version 12.10

IBM Informix Backup and Restore Guide

SC27-4508-02

Informix Product Family
Informix
Version 12.10

IBM Informix Backup and Restore Guide

SC27-4508-02

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

Edition

This edition replaces SC27-4508-01.

Contents

Part 5. archecker table level restore utility

Part 6. Backup and restore configuration parameter reference

Introduction

About this publication

These topics describe 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.

Types of users

These topics were written for the following users:

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

These topics are 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
- 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

You can access the Informix information centers, as well as other technical information such as technotes, white papers, and IBMRedbooks® publications online at http://www.ibm.com/software/data/sw-library/.

Software dependencies

This publication is written with the assumption that you are using IBM Informix Version 12.10 as your database 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 that is used by a language within a given territory and encoding is brought together in a single environment, called a Global Language Support (GLS) locale.

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

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

The examples in this publication are written with the assumption that you are using one of these locales: en_us.8859-1 (ISO 8859-1) on UNIX platforms or en_us.1252 (Microsoft 1252) in Windows environments. These locales support U.S. English format conventions for displaying and entering date, time, number, and currency values. They also support the ISO 8859-1 code set (on UNIX and Linux) or the Microsoft 1252 code set (on Windows), which includes the ASCII code set plus many 8-bit characters such as é, è, and ñ.

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

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

Demonstration databases

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

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

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

The scripts that you use to install the demonstration databases are in the `$INFORMIXDIR/bin` directory on UNIX platforms and in the `%INFORMIXDIR%\bin` directory in Windows environments.

What's New in the Backup and Restore Guide, Version 12.10

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

The following changes and enhancements are relevant to this publication. For a complete list of what's new in this release, go to http://pic.dhe.ibm.com/infocenter/informix/v121/topic/com.ibm.po.doc/new_features_ce.htm.

Table 1. What's New in IBM Informix Backup and Restore Guide for Version 12.10xC2

Overview	Reference
Restore critical files You can restore the following critical files that you backed up with the ON-Bar utility: • The `onconfig` file • UNIX: The `sqlhosts` file • The ON-Bar emergency boot file: `ixbar`.*servernum* • The server boot file: `oncfg_`*servername*.*servernum* Run the **onbar -r** command with the **-cf yes** option to restore critical files during a cold restore. Run the **onbar -r -cf only** command to restore only the critical files. Previously, you restored critical files by using storage manager commands.	"onbar -r syntax: Restoring data" on page 6-2
Optimize backups for data deduplication If your storage manager is enabled for data deduplication, use the IFX_BAR_USE_DEDUP environment variable to make backup operations more efficient. The new environment variable optimizes the format of backup images for deduplication processes.	"IFX_BAR_USE_DEDUP environment variable" on page 17-15
Enhanced support for IBM Tivoli® Storage Manager features If you use IBM Tivoli Storage Manager (TSM) with the ON-Bar utility to back up and restore data, you can configure ON-Bar to support the following TSM features: • You can speed back up and restore processes by setting a longer transfer buffer size. Set the IFX_BAR_NO_LONG_BUFFERS environment variable to prevent long transfer buffers. • You can replicate or import and export backup objects between TSM servers. Set the IFX_TSM_OBJINFO_OFF environment variable to prevent this feature.	"Configuring ON-Bar for optional TSM features" on page 4-6
ON-Bar activity log timestamps When a storage manager process hangs, the timestamp for the process in the ON-Bar activity log is inaccurate. The inaccurate timestamp represents the time at which the storage manager process started hanging instead of the current time. Inaccurate timestamps in the ON-Bar activity log are identified with an asterisk. The accompanying message lists the number of minutes after the timestamp that the storage manager process hung.	"Message format in the ON-Bar message log" on page 10-1

Table 2. What's New in IBM Informix Backup and Restore Guide for Version 12.10xC1

Overview	Reference
Enhanced built-in storage management for backup and restore IBM Informix Primary Storage Manager, which replaces IBM Informix Storage Manager (ISM), is easier to set up and use, even in embedded environments. You use the Informix Primary Storage Manager **onpsm** utility to manage storage for ON-Bar backup and restore operations, including parallel backups, that use file devices (disks). The **onsmsync** utility provides new commands that you can use to export backups to, and import them from, Informix Primary Storage Manager external device pools.	Chapter 15, "Informix Primary Storage Manager," on page 15-1 "The onsmsync utility" on page 8-4 "Informix Primary Storage Manager configuration parameters" on page 17-29

Example code conventions

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

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

```
CONNECT TO stores_demo
...

DELETE FROM customer
   WHERE customer_num = 121
...

COMMIT WORK
DISCONNECT CURRENT
```

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

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

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

Additional documentation

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

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

Compliance with industry standards

IBM Informix products are compliant with various standards.

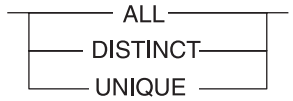
IBM Informix SQL-based products are fully compliant with SQL-92 Entry Level (published as ANSI X3.135-1992), which is identical to ISO 9075:1992. In addition, many features of IBM Informix database servers comply with the SQL-92 Intermediate and Full Level and X/Open SQL Common Applications Environment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

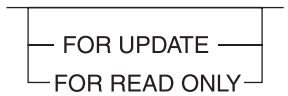
Syntax diagrams

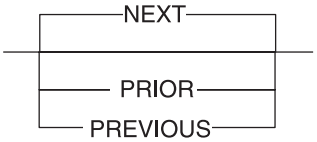
Syntax diagrams use special components to describe the syntax for statements and commands.

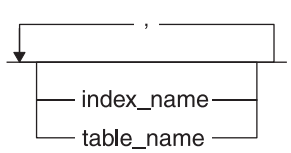
Table 3. Syntax Diagram Components

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. Only one item must be present.

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

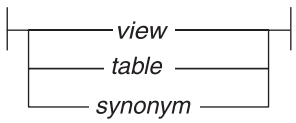
	`    .---NEXT---------.`
	`----+----------------+---`
	`    +---PRIOR--------+`
	`    '---PREVIOUS-----'`
		The values below the main line are optional, one of which you might specify. If you do not specify an item, the value above the line is used by default.

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

	`>>-| Table Reference |-><`	Reference to a syntax segment.

Table 3. Syntax Diagram Components (continued)

Component represented in PDF	Component represented in HTML	Meaning

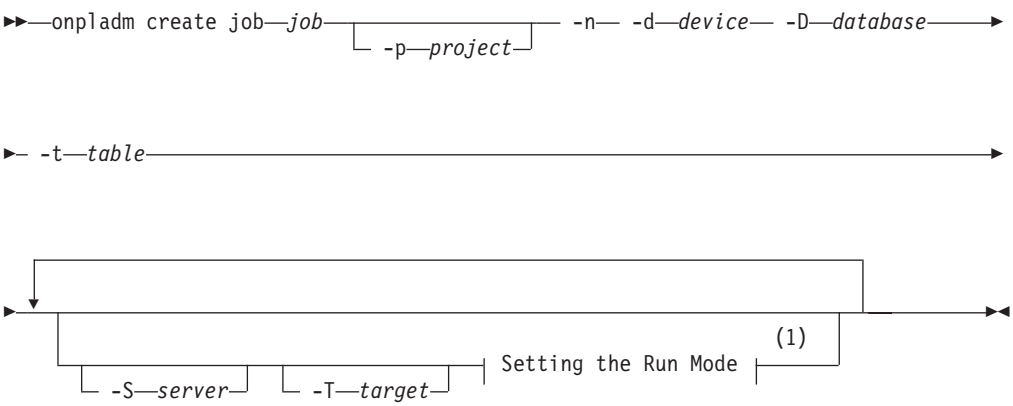
Table Reference
 
	Table Reference
	`|--+-----view--------+--|`
	`  +------table------+`
	`  '----synonym------'`
		Syntax segment.

How to read a command-line syntax diagram

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

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

Creating a no-conversion job

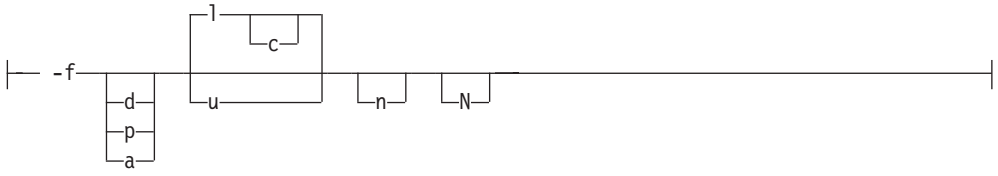


Notes:

1 See page Z-1

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

Setting the run mode:



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

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

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

Keywords and punctuation

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

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

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

Identifiers and names

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

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

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

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

How to provide documentation feedback

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

Use one of the following methods:

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

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

We appreciate your suggestions.

Part 1. Overview of backup and restore

These topics provide an overview of backup and restore concepts. They also provide information about planning for backup and restore operations.

Chapter 1. Backup and restore concepts

IBM Informix provides two utilities for backing up and restoring database server data. Both utilities back up and restore storage spaces and logical logs. However, they support different features and it is important to know the differences. These topics explain basic backup and restore concepts for IBM Informix database servers and compares the ON-Bar and **ontape** utilities.

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

Recovery system

A *recovery system*, which includes backup and restore systems, enables you to back up your database server data and later restore it if your current data becomes corrupted 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. You can also back up *blobspaces* and *sbspaces*.

The backup copy is typically written to a *secondary storage* medium such as disk or magnetic tape. 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 IBM Informix database files.

The following figure illustrates the basic concept of a database backup.

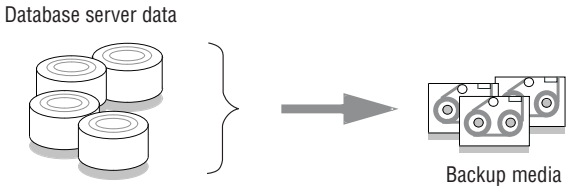


Figure 1-1. A backup of database server data

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.

Backup levels

To provide flexibility, the ON-Bar and **ontape** utilities support three backup levels.

Level 0

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-0 backups can be time-consuming because ON-Bar writes all the disk pages to back up 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

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 pages 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.

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.

Level 2

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.

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.

Important: If disks and other media are 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.

Related reference:

"Schedule backups" on page 2-2

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*.

Restriction: 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

You can manually back up logical logs or you can enable continuous logical-log backup.

A *manual logical-log backup* backs up all the full logical-log files and stops at the current logical-log file. You must monitor your logical logs carefully and start logical-log backups as needed.

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 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.

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. You can start continuous logical log backups by setting the ALARMPROGRAM configuration parameter in the `onconfig` file or by running an ON-Bar or **ontape** command.

Log salvage

When the database server is offline, you can perform a special 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.

Save logical-log backups

You should perform frequent logical-log backups and then save the logical-log backups from at least the last two level-0 backups so that you can use them to complete a restore.

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

You should save the logical-log backups from the last two level-0 backups because if a level-0 backup is inaccessible or unusable, you can restore data from an older

backup. 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.

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

To illustrate, as the following figure shows, suppose you perform a level-0 backup on Monday at 10 p.m. and then back up the logical logs on Tuesday at midnight. On Wednesday at 11 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 a.m. on Wednesday unless you had continuous logical-log backup setup.

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.

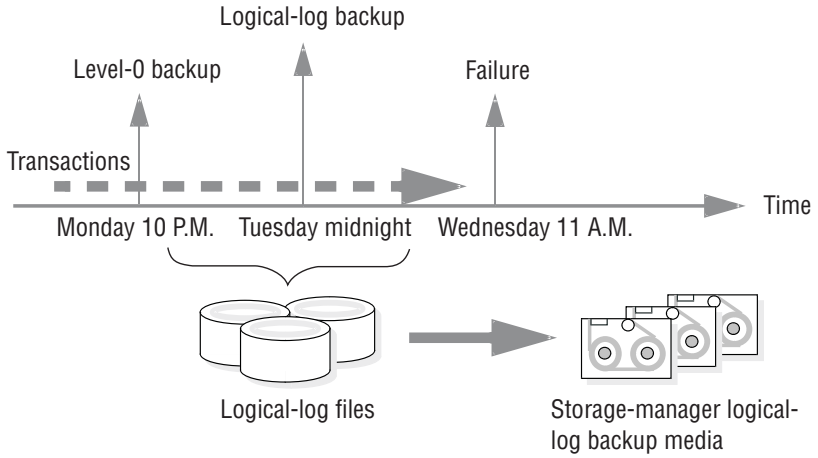


Figure 1-2. Storage space and logical-log backups

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.

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.

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. The following figure illustrates a physical restore.

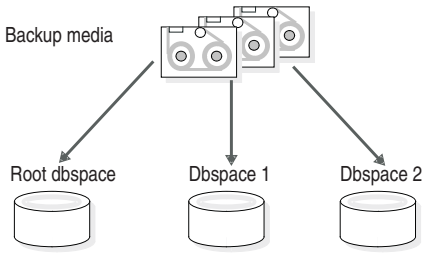


Figure 1-3. Physical restore

Logical restore

As the following figure 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.

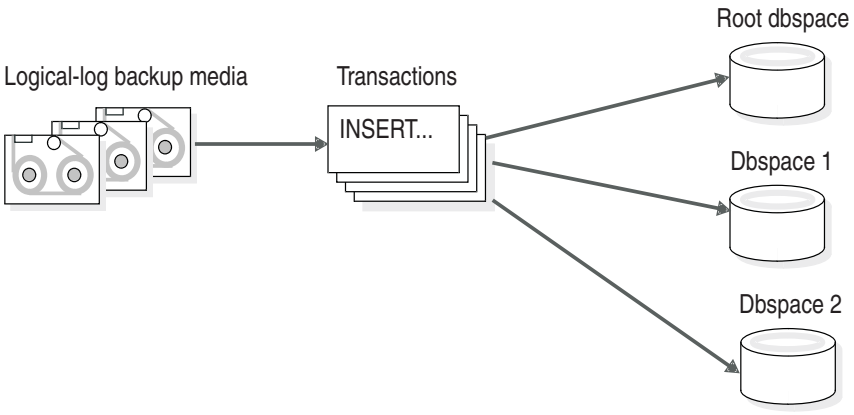


Figure 1-4. Logical restore

The database server automatically knows which logical logs to restore.

For more information, see Chapter 6, "Restore data with ON-Bar," on page 6-1 and Chapter 13, "Restore with ontape," on page 13-1.

Warm, cold, and mixed restores

When you restore data, you must decide whether to do so while the database server is in quiescent, online, or offline mode. The type of restore depends on which of these operating modes the server is in.

The types of restores are as follows:

- If you restore noncritical dbspaces while the database server is online or quiescent, that process is called a *warm restore*.
- When IBM Informix is offline, you can perform only 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 the following figure 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.

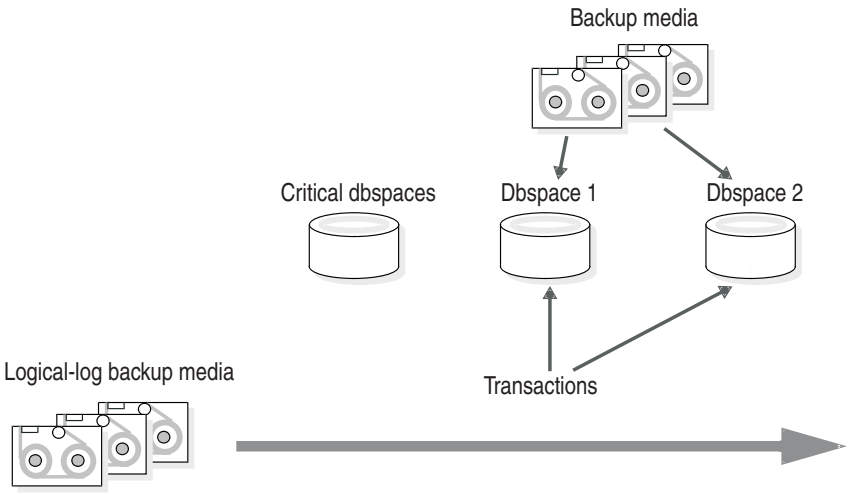


Figure 1-5. Warm restore

You cannot perform more than one simultaneous warm restore.

Cold restore

As the following figure 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.

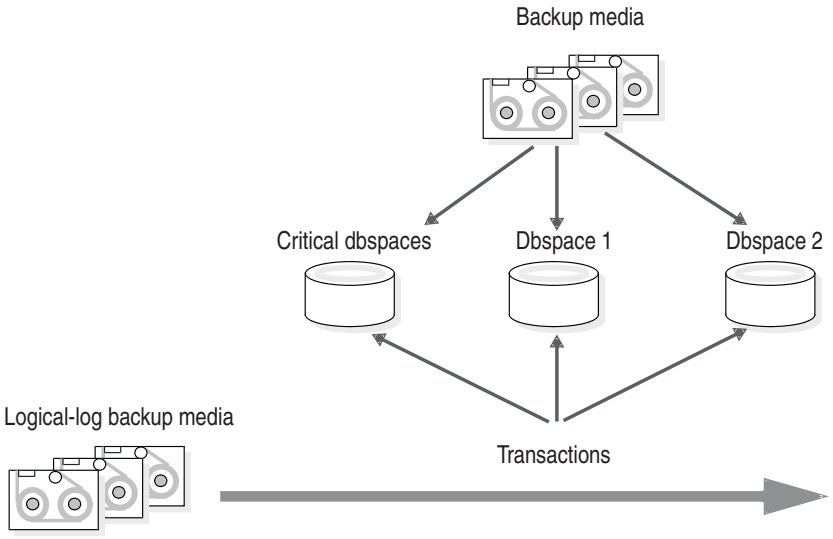


Figure 1-6. Cold restore

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.

When restoring a whole-system backup, it is not necessary to restore the logical logs. A whole-system backup contains a snapshot of the entire instance at the moment the backup was performed, which is logically consistent across all dbspaces.

When restoring a standard backup, you must restore the logical logs by performing a logical restore.

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.

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.

Required: Cold restores are required for Enterprise Replication servers before resuming replication.

Mixed restores

A mixed restore makes the critical data available sooner, however, 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.

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.

Related reference:

"Determine failure severity" on page 2-1

Continuous log restore

A *continuous log restore* keeps a second system available to replace the primary system if the primary system for restoring logs fails.

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 clean up 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 a continuous log restore by using ON-Bar" on page 6-10 and "Configuring continuous log restore with ontape" on page 13-10.

Related tasks:

"Configuring a continuous log restore by using ON-Bar" on page 6-10

"Configuring continuous log restore with ontape" on page 13-10

Comparison of the ON-Bar and ontape utilities

This topic contains information to help you compare the ON-Bar and **ontape** utilities, so you can determine when to use each utility.

ON-Bar

Backs up and restores storage spaces (dbspaces) and logical files, by 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:

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

Important: The backup 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.

The following table compares ON-Bar and **ontape**.

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?	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

Table 1-1. Differences between ON-Bar and ontape (continued)

Can the utility...	ON-Bar	ontape
Transform data with external programs?	yes	yes
Back up to or restore from cloud storage?	no	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 IBM Informix Primary Storage Manager, supports simultaneous sessions.
- Device support and storage management

  The **ontape** utility supports remote backup devices on other hosts.

  ON-Bar, with the Informix Primary Storage Manager, supports the export of backup generations into specified directories and devices.

  You can also use ON-Bar with the Tivoli storage manager or third-party storage managers to obtain 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 to change the logging mode.

For details about each utility, see Chapter 5, "Back up with ON-Bar," on page 5-1 and Chapter 12, "Back up with ontape," on page 12-1.

Related reference:

Chapter 11, "Configure ontape," on page 11-1

Chapter 2. Plan for backup and restore

These topics describe the planning for backup and restore, for example by planning your recovery strategy and backup system.

Plan a recovery strategy

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

Types of data loss

The first step in planning a recovery strategy 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

Determine failure severity

After you determine your recovery goals, create your recovery plan. The plan should include recovery goals for multiple levels of failure.

The following table shows recovery plans for failures with amounts of lost data.

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.

Related concepts:

"Warm, cold, and mixed restores" on page 1-5

Data use determines your backup schedule

After you develop your recovery plan, create a backup plan based on how you use your data.

How you use the data 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 downtime where the system can be restored?
- If your system is 24x7 (no downtime), 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?

Schedule backups

You recovery strategy should include a schedule of backups. 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.

Your backup plan should also specify the backup level.

The following table shows a sample backup plan for a small or medium-sized system.

Table 2-2. Sample backup plan

Backup level	Backup schedule
Complete backup (level-0)	Saturday at 6 p.m.
Incremental backup (level-1)	Tuesday and Thursday at 6 p.m.
Incremental backup (level-2)	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 "Preparing to back up data" on page 5-1.)

Related concepts:

"Backup levels" on page 1-1

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 an additional privilege to back up or restore data.

LBAC protection remains intact after you restore data with ON-Bar or **ontape**.

Plan 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.

Actions after which to perform a level-0 back up

You must perform a level-0 backup of, at minimum, the root dbspace and the modified storage spaces after you perform any of the following actions:

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

For example, if you add a new dbspace **dbs1**, you 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.

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

If you create a 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 storage space with the same name.
2. After you create the storage space, back up the root dbspace and the new storage space.

Actions before which to perform a level-0 back up

You must perform a level-0 backup of the modified storage spaces before you perform any of the following actions:

- Convert a nonlogging database to a logging database.
- Before you alter a RAW table to type STANDARD. This backup ensures that the unlogged data is restorable before you switch to a logging table type.

Related reference:

"onbar -b syntax: Backing up" on page 5-2

Evaluate hardware and memory resources

When planning your backup system, evaluate your 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

Also consider temporary disk space needed for backup and restore. The database server uses temporary disk space to store the before images of data that are overwritten while backups are occurring and overflow from query processing that occurs in memory.

When preparing to back up data, make sure that you correctly set the DBSPACETEMP environment variable or parameter to specify dbspaces with enough space for your needs. If there is not enough room in the specified dbspaces, the backup will fail, root dbspace will be used, or the backup will fail after filling the root dbspace.

Evaluate backup and restore time

Several factors. including database server configuration and the size of your database, affect the amount of time that the system needs to back up and restore data.

How long your backup or restore takes depends on the following factors:

- 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 storage spaces 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 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 configuration parameters to control the number and size of data buffers.

Evaluate logging and transaction activity

When planning your backup system, also consider logging and transaction activity.

The following database server usage requirements 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 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.

Compress row data

Compressing row data can make backing up and restoring data more efficient.

Compressing row data before backing it up can improve the speed of backing up and restoring and requires less backup media. A smaller size of data results in the following advantages over uncompressed data during backup and restore:

- Backing up is quicker.
- Restoring is quicker.
- The logical logs are smaller.
- The backup image is smaller.

Using an external compression utility to compress a backup image of compressed row data might not reduce the size of the backup image, because already compressed data usually cannot be further compressed. In some cases, the size of the backup image of compressed row data might be larger than the size of the backup image that was compressed by an external utility.

Transform data with external programs

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

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

Tip: If you compress row data before backing it up, compressing the backup image with an external utility might not result in a smaller backup image.

The filter can be owned by anyone, but cannot have write access to non-privileged users. Permission on the filters is the same as that of permission on any other executable file that is called by an IBM Informix server or an Informix utility.

Part 2. ON-Bar backup and restore system

Related reference:

Chapter 17, “Backup and restore configuration parameters,” on page 17-1

Chapter 3. Overview of the ON-Bar backup and restore system

ON-Bar consists of various components and it works with a storage manager to back up and restore data.

ON-Bar components

ON-Bar components include a command-line utility, catalog tables, an activity log, and an emergency boot file. You use ON-Bar with a storage manager and the XBSA shared library for the storage manager.

The following figure shows the ON-Bar and database server components:

- The storage spaces (dbspaces, blobspaces, and sbspaces) and logical logs from the database server
- The **sysutils** database, which includes ON-Bar catalog tables
- The **onbar** and the **onbar-d** command-line utilities
- The XBSA shared library for the storage manager on your system
- The storage media for storing backups
- The ON-Bar activity log
- The ON-Bar emergency boot file

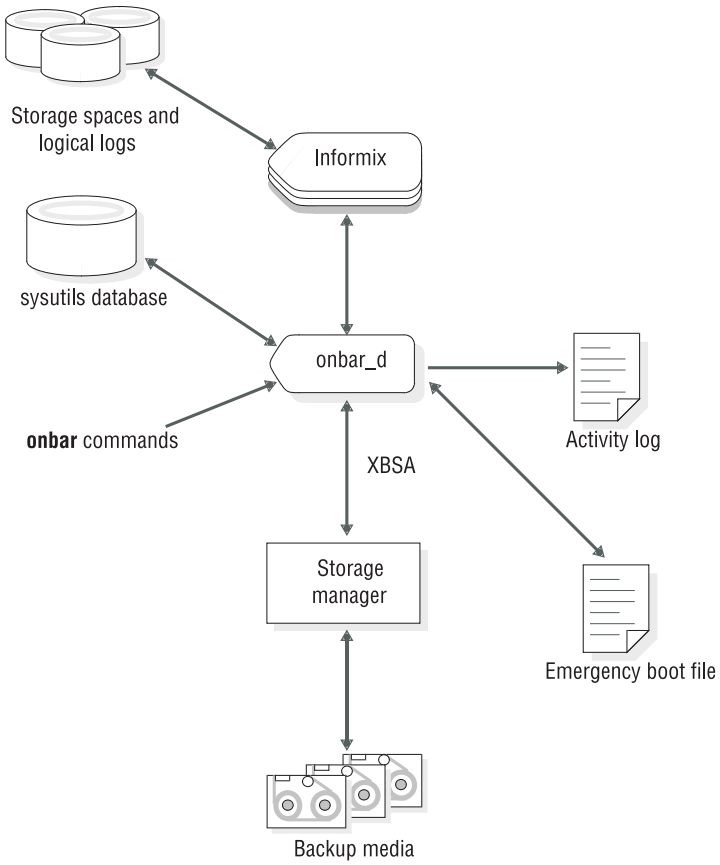


Figure 3-1. ON-Bar components for Informix

ON-Bar communicates with both the database server and the storage manager. You use the **onbar** command to start a backup or restore operation. By default, ON-Bar backs up and restores storage spaces in parallel. ON-Bar always processes log files serially.

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.

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 also places the following critical files in the archive during backups:

- The `onconfig` file
- UNIX: The `sqlhosts` file
- The ON-Bar emergency boot file: `ixbar.`*servernum*
- The server boot file: `oncfg_`*servername.servernum*

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.

ON-Bar status and error messages are written to the activity log file: `bar_act.log`.

Backup Services API (XBSA)

ON-Bar and the storage manager communicate through the Backup Services Application Programming 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 various 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 IBM Informix Primary Storage Manager, you must also use the XBSA shared library provided with ON-Bar. ON-Bar and the XBSA shared library must be compiled the same way (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 in the correct order to the correct instance of the database server, 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.

Related concepts:

Chapter 15, “Informix Primary Storage Manager,” on page 15-1

ON-Bar catalog tables

ON-Bar uses the catalog tables in the **sysutils** database to track backup and restore operations. The **onsmsync** utility uses other catalog tables to track its operations.

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 and maintains the following tables to track its operations:

- The **bar_ixbar** table contains the history of all unexpired successful backups in all timelines.
- The **bar_syncdeltab** table is normally empty except when **onsmsync** is running.

For a description of the content of these tables, see Chapter 9, "ON-Bar catalog tables," on page 9-1.

ixbar file: ON-Bar emergency boot file

The emergency boot file is automatically updated after every backup. It contains the information that ON-Bar needs to perform a cold restore.

Important: Do not modify the emergency boot file. Doing so might cause ON-Bar to select the wrong backup as part of a restore, possibly leading to data corruption or system failure.

The file name for the boot file is `ixbar.`*`servernum`*, where *servernum* is the value of the SERVERNUM configuration parameter.

The ON-Bar emergency boot file is in the `$INFORMIXDIR/etc` directory on UNIX and in the `%INFORMIXDIR%\etc` directory on Windows. You can override the default path and name of the boot file by changing the information specified in the BAR_IXBAR_PATH configuration parameter.

bar_act.log file: ON-Bar activity log

ON-Bar writes informational, progress, warning, error, and debugging messages to the ON-Bar activity log, `bar_act.log`.

ON-Bar backup and restore errors do not appear in standard output. If an error occurs when you back up and restore data, check information in the ON-Bar activity log

You can also 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

The ON-Bar activity log is in the `/tmp` directory on UNIX and in the `%INFORMIXDIR%\etc` directory on Windows. You specify the location of the ON-Bar activity log with the BAR_ACT_LOG configuration parameter.

Related reference:

"BAR_ACT_LOG configuration parameter" on page 17-3

Chapter 10, "ON-Bar messages and return codes," on page 10-1

"View ON-Bar backup and restore performance statistics" on page 8-10

"onbar -m syntax: Monitoring recent ON-Bar activity" on page 5-8

ON-Bar script

The ON-Bar utility includes a shell script on UNIX and a batch script on Windows for customizing backup and restore operations.

When you install ON-Bar with the database server, a default script is included. The name and location of the script depends on the operating system:

UNIX The **onbar** shell script is in the `$INFORMIXDIR/bin` directory.

Windows
The **onbar.bat** batch script is in the `%INFORMIXDIR%\bin` directory.

When you issue ON-Bar commands from the command line, the arguments are passed to the script, and then to the **onbar_d** utility.

Table 3-1. ON-Bar utilities

Utility	Description
onbar_d utility	Transfers data between the database server and the storage manager. The **onbar** command calls the **onbar_d** utility that starts the **onbar-driver**. The **onbar-driver** starts and controls backup and restore activities.
onsmsync utility	Synchronizes the contents of the **sysutils** database, the emergency boot files, and the storage manager catalogs. Use this utility to purge backups that are no longer needed.
ondblog utility	Changes the database-logging mode. The **ondblog** utility logs its output in the ON-Bar activity log, `bar_act.log`.
archecker utility	Verifies backups, and restores table-level data from an archive.

Related concepts:

"Overview of the archecker utility" on page 16-1

Related reference:

"The onsmsync utility" on page 8-4

Chapter 4. Configure the storage manager and ON-Bar

The topics in this section provide the information that you need to plan and to set up ON-Bar with a storage manager.

Related tasks:

"Preparing to back up data" on page 5-1

Configure a storage manager

ON-Bar backup and restore operations require a storage manager that integrates with ON-Bar through an XBSA shared library interface.

You can choose to use the IBM Informix Primary Storage Manager, the IBM Tivoli Storage Manager (TSM), or a third-party storage manager with ON-Bar. The Informix Primary Storage Manager is bundled with Informix. If you are using TSM, the XBSA shared library needed for ON-Bar to communicate with TSM is bundled with Informix.

The Informix Primary Storage Manager manages storage for ON-Bar backup and restore operations, including parallel backups, that use file devices (disks) only, not tapes. By default, the IBM Informix Primary Storage Manager is automatically configured with the information specified in Informix Primary Storage Manager and some ON-Bar configuration parameters. This storage manager is also automatically configured when you use the **onpsm** utility. You can change the configuration. For information, see Chapter 15, "Informix Primary Storage Manager," on page 15-1 and"Configuring Informix Primary Storage Manager" on page 15-9

Related concepts:

Chapter 15, "Informix Primary Storage Manager," on page 15-1

"Setting up Informix Primary Storage Manager" on page 15-8

Storage-manager definitions in the sm_versions file

Most storage managers must have an entry in the `sm_versions` file.

The IBM Informix Primary Storage Manager and Tivoli Storage Manager do not require an entry in the `sm_versions` file.

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

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

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

Before ON-Bar starts a backup or restore process with the Tivoli Storage Manager and third-party storage managers, ON-Bar 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.

Related tasks:

"Updating the storage-manager definition in the sm_versions file for TSM" on page 4-5
"Configuring a third-party storage manager" on page 4-6

Configuring TSM

To use IBM Tivoli Storage Manager (TSM) with IBM 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.

You must also configure IBM Informix Interface for TSM and perform other TSM configuration tasks on your IBM Informix database server computer.

To configure TSM:

1. Edit the TSM client options files.
2. Assign a TSM management class for the server to use for backups.
3. Set the IBM Informix Interface for TSM environment variables.
4. Register with the TSM server.
5. Initialize the IBM Informix Interface for TSM.
6. Optional. Configure ON-Bar to support optional TSM features.

The version of the XBSA shared library for TSM is 1.0.3.

For details 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*

Editing the TSM client options files

The IBM Informix Interface for Tivoli Storage Manager (TSM) communicates with the TSM server with the TSM API. By default, IBM Informix Interface for TSM uses the client user options file (`dsm.opt`) and, on UNIX systems, the client system options file (`dsm.sys`), both of which are located in the TSM API installation directory.

On UNIX systems, edit both the `dsm.opt` and the `dsm.sys` files as the **root** user:

- 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.

On Windows systems, specify the TMS server name, communication method, and server options in the `dsm.opt` file.

The following example shows an example `dsm.opt` file on Windows:

```
SErvername  PAX12_TSMSERVER1
  COMMMethod        TCPip
  TCPPort            1500
  TCPADMINPort         1500
  TCPServeraddress  9.25.148.226
  PasswordAccess    generate
```

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:

You can edit the IBM Tivoli Storage Manager (TSM) client user options file, `dsm.opt`. This file 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
Identifies which TSM server instance, as listed in the `dsm.sys` file, that IBM Informix Interface for TSM contacts for services.

TRACEFILE
Sends trace output information to a designated file.

TRACEFLAG
Sets specific trace flags

Editing the TSM client system options file:

You can edit the IBM Tivoli Storage Manager (TSM) client systems options file, `dsm.sys`. This file 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
Specifies the name that 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
Identifies the communication method.

TCPSERVERADDRESS
Identifies the TSM server.

PASSWORDACCESS
Specifies 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 contacted.

You can enable deduplication by including the DEDUP=CLIENTORSERVER option in the client system options file. You must also set the **IFX_BAR_USE_DEDUP** environment variable in the Informix environment and restart the database server. See the *Tivoli Storage Manager Backup-Archive Client Installation and User's Guide* for information about configuring deduplication.

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.

Related concepts:

"Configuring ON-Bar for optional TSM features" on page 4-6

Related reference:

"IFX_BAR_USE_DEDUP environment variable" on page 17-15

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.

The INCLUDE 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 backup, 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.

Setting the IBM Informix Interface for TSM environment variables

When you use the IBM Informix Interface for TSM, you need to set certain environment variables in the environment of the user.

The following table describes these environment variables.

Table 4-1. IBM Informix 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.
DSMI_DIR	On UNIX, 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. On Windows, specifies the installation location of the TSM Backup-Archive Client. Typically, the TMS Backup-Archive Client is installed in the `C:\Tivoli\TSMClient\baclient` directory.
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 user **informix** or the backup operator should have write permission on this directory.

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
```

The following example shows how to set up these environment variables for Windows if the TSM API is installed in the C:\Tivoli\TSMClient\api directory:

```
set DSMI_CONFIG=C:\Tivoli\TSMClient\api\BIN\dsm.opt
set DSMI_DIR=C:\Tivoli\TSMClient\baclient
set DSMI_LOG=C:\logdir
```

Registering with the TSM server

Before backing up to and recovering from an IBM Tivoli Storage Manager (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 IBM Informix Interface for TSM node is registered with a TSM server, you can begin using the IBM Informix Interface for TSM to back up and restore your IBM 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 IBM 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 IBM Informix Interface for TSM password

To initialize the password for IBM 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 IBM 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 storage-manager definition in the sm_versions file for TSM

You must update the storage-manager definition in sm_versions file for ON-Bar to use with IBM Tivoli Storage Manager (TSM).

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 file:

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. Put tsm in the **sm_name** field of the sm_versions file. The value adsm is also valid but will be deprecated in a future release.

3. 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.

The following example shows the Tivoli Storage Manager definition in the `sm_versions` file:

```
1|5.3|tsm|5
```

Related reference:

"Storage-manager definitions in the sm_versions file" on page 4-1

Configuring ON-Bar for optional TSM features

You can configure ON-Bar to enable or disable optional features in IBM Tivoli Storage Manager (TMS).

Apply all patches and updates to TSM before you use the following TSM features.

Deduplication

Deduplication eliminates redundant data in backups. To enable Informix support for deduplication, set the **IFX_BAR_USE_DEDUP** environment variable in the Informix environment and restart the database server. Update the TSM client systems option file.

Large transfer buffer size

The default transfer buffer size is 64 KB. Set the BAR_XFER_BUF_SIZE configuration parameter to specify a transfer buffer of up to 65 MB.

To limit the transfer buffer size to 64 KB regardless of the value of the BAR_XFER_BUF_SIZE configuration parameter, set the **IFX_NO_LONG_BUFFERS** environment variable to 1.

Replicate, import, and export backup objects

Replicating, importing, or exporting backup objects between TSM servers requires unique IDs for backup objects. ON-Bar automatically stores IDs in the metadata of backup objects that are unique for all TSM server with version 12.10.xC2 or later.

To disable the ability to restore backup objects that are moved between TSM servers, set the **IFX_TSM_OBJINFO_OFF** environment variable to 1.

Related reference:

"IFX_BAR_USE_DEDUP environment variable" on page 17-15

"BAR_XFER_BUF_SIZE configuration parameter" on page 17-13

"Editing the TSM client system options file" on page 4-3

"IFX_TSM_OBJINFO_OFF environment variable" on page 17-16

"IFX_BAR_NO_LONG_BUFFERS environment variable" on page 17-15

Configuring a third-party storage manager

Storage managers have slightly different installation and configuration requirements. If you use a third-party storage manager, make sure that you follow the manufacturer instructions carefully. If you have difficulty with the storage-manager installation and configuration, contact the manufacturer directly.

For the list of certified storage managers for your ON-Bar version, consult your marketing representative.

Important: Some storage managers let you specify the 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 by following the instructions in your storage-manager documentation. The storage manager must know the device names of the storage devices that it uses.
4. Label your storage volumes.
5. Mount the storage volumes on the storage devices.
6. Create the storage-manager definition in the `sm_versions` file. Use the definition provided by the vendor of the third-party storage manager.
   a. 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.
   b. Create your own `sm_versions` file with the correct data for the storage manager by using the format in `sm_versions.std` as a template. To find out which code name to use in `sm_versions` for third-party storage managers, see the storage-manager documentation.
   c. 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.
7. Verify that the BAR_BSALIB_PATH configuration parameter points to the correct XBSA shared library for your storage manager.
8. If you enabled deduplication for your storage manager, set the IFX_BAR_USE_DEDUP environment variable and restart the database server.
9. ON-Bar uses the value of the SERVERNUM configuration parameter as part of the storage path for the logical logs in the storage manager. If the storage manager does not use a wildcard for the server number, set the appropriate server number environment variable for the storage manager.

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.

Related reference:

"IFX_BAR_USE_DEDUP environment variable" on page 17-15

"Storage-manager definitions in the sm_versions file" on page 4-1

Validating your storage manager

When you convert or revert an IBM Informix database server, the storage manager that you used on the old version might not be validated for the version that you are migrating to. Verify that the storage-manager vendor successfully completed the IBM 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

Before you begin your first backup, review the default ON-Bar parameters in the `onconfig` file and adjust the values as needed. You can also set an environment variable.

You can configure the behavior of ON-Bar by setting the following configuration parameters and environment variable.

Table 4-2. ON-Bar configuration parameters and environment variable

Behavior	Configuration parameters
Control the number and size of data buffers and the number of parallel processes.	"BAR_NB_XPORT_COUNT configuration parameter" on page 17-9 "BAR_XFER_BUF_SIZE configuration parameter" on page 17-13 "IFX_BAR_NO_LONG_BUFFERS environment variable" on page 17-15 "BAR_MAX_BACKUP configuration parameter" on page 17-8
Set the debugging level and the location of debug log file.	"BAR_DEBUG configuration parameter" on page 17-5 "BAR_DEBUG_LOG configuration parameter" on page 17-7
Change the path of the ON-Bar boot file.	"BAR_IXBAR_PATH configuration parameter" on page 17-8
Maintain a history of expired backups.	"BAR_HISTORY configuration parameter" on page 17-7
Change the location and contents of ON-Bar activity log.	"BAR_IXBAR_PATH configuration parameter" on page 17-8
Maintain a history of expired backups.	"BAR_ACT_LOG configuration parameter" on page 17-3 "BAR_PROGRESS_FREQ configuration parameter" on page 17-11 "BAR_PERFORMANCE configuration parameter" on page 17-10
Set automatic retrying of failed back ups or restores.	"BAR_RETRY configuration parameter" on page 17-11
Allow a failed restore to be restarted.	"RESTARTABLE_RESTORE configuration parameter" on page 17-19
Increase backup size estimate sent to the storage manager.	"BAR_SIZE_FACTOR configuration parameter" on page 17-12
Extend the time an RS secondary server waits for a checkpoint during an external backup.	"BAR_CKPTSEC_TIMEOUT configuration parameter" on page 17-5
Configure continuous log backup.	ALARMPROGRAM

Table 4-2. ON-Bar configuration parameters and environment variable (continued)

Behavior	Configuration parameters
Filter or transform backed up data with an external program.	"BACKUP_FILTER configuration parameter" on page 17-2 "RESTORE_FILTER configuration parameter" on page 17-20
Optimize the deduplication capabilities of storage managers.	"IFX_BAR_USE_DEDUP environment variable" on page 17-15
Disable the ability to replicate, import, or export backup objects among TSM servers.	"IFX_TSM_OBJINFO_OFF environment variable" on page 17-16
Force the use of the `sm_versions` file.	"IFX_BAR_NO_BSA_PROVIDER environment variable" on page 17-14

Do not set the LTAPEDEV configuration parameter to `/dev/null` or `NUL` because logical-log backups would be disabled and you can restore only whole-system backups.

ON-Bar security

By default, only the **informix** or **root** users on UNIX system or members of the **Informix-Admin** group on Windows systems can run ON-Bar commands.

To enable additional users to run ON-Bar commands:

- On UNIX systems, create a **bargroup** group and add users to the group. For instructions on how to create a group, see your UNIX documentation.
- On Windows systems, add the users to the **Informix-Admin** group.

Restriction: For security, it is recommended that ON-Bar commands not be run by the **root** user.

Related reference:

"onbar -r syntax: Restoring data" on page 6-2

"onbar -b syntax: Backing up" on page 5-2

"onbar -v syntax: Verifying backups" on page 5-12

"onbar -m syntax: Monitoring recent ON-Bar activity" on page 5-8

"onbar -P syntax: Printing backed-up logical logs" on page 5-10

"onbar -RESTART syntax: Restarting a failed restore" on page 6-18

Verifying the configuration of ON-Bar and your storage manager

Before you begin using ON-Bar and your storage manager, make sure that ON-Bar and your storage manager are set up correctly.

Verify your configuration by checking the items in the following list:

- The storage manager is installed and configured to manage specific storage devices.
- For UNIX, 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.
- For Windows, make sure that the BAR_BSALIB_PATH configuration parameter specifies correctly the XBSA shared library.

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

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, "Back up with ON-Bar," on page 5-1.

Files that ON-Bar and storage managers use

ON-Bar, IBM Informix Primary Storage Manager, and IBMTivoli Storage Manager (TSM) use particular files in your installation.

The following table lists the files that ON-Bar and IBM Tivoli Storage Manager (TSM) use and the directories where the files are. These names and locations change if you set up the onconfig file to values different from the defaults.

Table 4-3. List of files that ON-Bar and TSM use

File name	Directory	Purpose
ac_config.std	UNIX: $INFORMIXDIR/etc Windows: %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	The **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 "bar_act.log file: ON-Bar activity log" on page 3-4.
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 "ixbar file: ON-Bar emergency boot file" on page 3-4.

Table 4-3. List of files that ON-Bar and TSM use (continued)

File name	Directory	Purpose
oncfg_*servername.servernum*	$INFORMIXDIR/etc	Configuration information for ON-Bar restores.
	%INFORMIXDIR%\etc	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.
save, savegrp, savefs	$INFORMIXDIR/bin	
sm_versions	$INFORMIXDIR/etc	Identifies the version of a third-party storage manager.
	%INFORMIXDIR%\etc	To update the storage-manager version, edit the sm_versions file directly. The Informix Primary Storage Manager does not use the **sm_versions.std** file.

Chapter 5. Back up with ON-Bar

You can use the ON-Bar utility to back up and verify storage spaces (dbspaces, blobspaces, and sbspaces) and logical-log files.

To perform a backup with ON-Bar:

1. Prepare for backup.
2. Back up with ON-Bar
3. Monitor backup progress.
4. Verify backups.
5. Back up storage manager information.

You can customize ON-Bar and storage manager commands in a shell or batch script. You can call ON-Bar from a job-scheduling program.

Related reference:

"Customizing ON-Bar and storage-manager commands" on page 8-1

Preparing to back up data

Before you back up storage spaces and logical logs, you must prepare the system and copy critical administrative files.

To prepare to back up data:

1. Configure ON-Bar and your storage manager.
2. Ensure 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.
3. Verify that you have enough temporary disk space. The database server uses temporary disk space to store the before images of data that are overwritten while backups are occurring and overflow from query processing that occurs in memory. Verify that the **DBSPACETEMP** environment variable and DBSPACETEMP configuration parameter specify dbspaces that have enough space for your needs. If there is not enough room in the specified dbspaces, the backup will fail, root dbspace will be used, or the backup will fail after filling the root dbspace.
4. Back up administrative files to a different location.
5. Run the **oncheck -cD** command to verify that all database server data is consistent. You do not need to check for consistency before every level-0 backup. Do not discard a backup that is known to be consistent until the next time that you verify the consistency of your databases.

Related reference:

oncheck -cd and oncheck -cD commands: Check pages (Administrator's Reference)

Chapter 4, "Configure the storage manager and ON-Bar," on page 4-1

"onbar -b syntax: Backing up" on page 5-2

Administrative files to back up

Although ON-Bar backs up some critical administrative files, you must also include critical files in normal operating-system backups of important configuration files.

Files that ON-Bar backs up

When you back up a storage space, ON-Bar also backs up the following critical files:

- The `onconfig` file
- UNIX: The `sqlhosts` file
- The ON-Bar emergency boot file: `ixbar.`*servernum*
- The server boot file: `oncfg_`*servername.servernum*

You must restore these files if you need to replace disks or if you restore to a second computer system (imported restore).

Look at the `bar_act.log` file to determine whether critical files are successfully backed up. The return code for the **onbar -b** command indicates only whether storage spaces are successfully backed up. The following lines from the `bar_act.log` file show that the ON-Bar emergency boot file, `ixbar.0`, is backed up:

```
Begin backup of critical file '/opt/informix-11.70.fc7/etc/ixbar.0'.
Completed backup of critical file '/opt/informix-11.70.fc7/etc/ixbar.0'
```

Files that you must manually back up

In addition to the critical files, you must also manually back up the following administrative files:

- The `sm_versions` file
- Storage-manager configuration and data files
- Simple-large-object data in blobspaces that are stored on disks
- Externally stored data such as external tables that a DataBlade® maintains

Tip: Even though ON-Bar includes the critical files with the files it backs up, it is a good practice to also include the critical files in your system archive. Having the critical files included in both the IBM Informix and system archives gives you more options if you need them.

Files that ON-Bar re-creates

Although ON-Bar does not back up the following items, ON-Bar automatically re-creates 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 re-creates empty temporary dbspaces.

onbar -b syntax: Backing up

Use the **onbar -b** command to back up storage spaces and logical logs.

To run ON-Bar commands, you must be user **root**, user **informix**, a member of the **bargroup** group on UNIX, or a member of the Informix-Admin group on Windows.

Syntax for backing up with ON-Bar

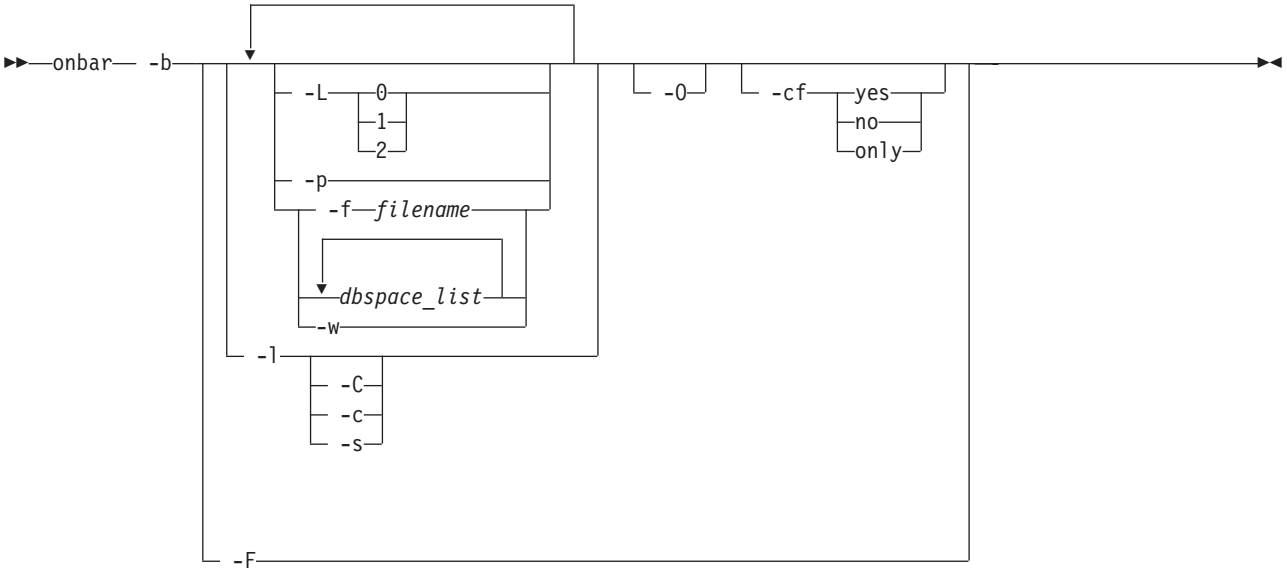


Table 5-1. Options for the onbar -b command

Option	Description
-b	Specifies a backup Backs up the storage spaces and logical logs, including the current logical log.
dbspace_list	Specifies the storage spaces to be backed up, separated by blank spaces. If you do not enter *dbspace_list* or `-f` *filename*, ON-Bar backs up all online storage spaces on the database server.
-c	Closes and backs up the current logical log and the other full logical logs.
-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, stop the ON-Bar process with an interrupt command, such as CTRL-C or SIGTERM.

Table 5-1. Options for the onbar -b command (continued)

Option	Description
-cf	Specifies whether the critical files are backed up. The critical files are the `onconfig` file, the `sqlhosts` file, and the `ixbar.`*servernum* file. Valid values are: • **yes** = Backs up the critical files. Default when performing a level 0, 1, or 2 backup. • **no** = Does not back up the critical files. Default when backing up the logical log files. • **only** = Backs up only the critical files.
-f *filename*	Backs up the storage spaces that are listed in the text file that is specified by the *filename* value. Use this option to avoid entering a long list of storage spaces every time that you back up. For more information, see "List of storage spaces in a file" on page 5-7.
-F	Performs a fake backup A storage-manager application is not necessary. No backup actually occurs, so no restore is possible from a fake backup. Use fake backups sparingly, if at all. Fake backups might be appropriate in the following situations: • Change database logging modes • Change a RAW table to a STANDARD table • Allow the user to use new logs, chunks, or mirrors without performing a backup • In special situations when you, the administrator, judge that a backup is not needed
-L	Specifies the level of backup to perform on storage spaces: • 0 = a complete backup (Default) • 1 = changes since the last level-0 backup • 2 = changes since the last level-1 backup If you request an incremental backup and ON-Bar finds that no previous level backup was 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.
-l	Performs a backup of full logical-log files. The current logical-log file is not backed up.
-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	Backs up only physical storage spaces without logical logs. A warning message is written to the activity log listing the log unique ID of the latest log file that is required for a restore of the storage spaces. Use this option if logical logs are being continuously backed up. If necessary, 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.

Table 5-1. Options for the onbar -b command (continued)

Option	Description
-s	Salvages any logical logs that are still on disk after a database server failure. You can run the **onbar -l -s** command while the server is offline. 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.
-w	Backs up a whole system, which includes all storage spaces and logical logs based on a single checkpoint. The time of the backup is stored with the backup information. The data in all storage spaces is consistent in a whole-system backup, therefore, you do not need to restore the logical logs to make the data consistent. If you do not save the logical logs, you must use the **-w** option.

Usage

Before you back up your data, make sure that your data is consistent by running the **oncheck -cD** command.

To run ON-Bar commands, you must be user **root**, user **informix**, or a member of the **bargroup** group on UNIX, or a member of the **Informix-Admin** group on Windows. For more information, see "ON-Bar security" on page 4-9.

You can back up storage spaces and logical logs when the database server is in online, quiescent, or fast-recovery mode.

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** command to determine which storage spaces are online. During a backup, if ON-Bar encounters a down dbspace, it skips it and later returns an error. If a storage space is offline, restart the backup when the storage space is back online.

After you begin the backup, monitor its progress in the ON-Bar activity log and database server message log.

You can either back up the logical logs separately or with storage spaces. 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 by running the **onbar -b -C** command. Logical-log backups are always level 0. After you close the current logical log, you can back it up.

If you perform whole-system backups and restores, you do not need to restore logical logs. However, 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.

If you are running continuous logical log backup and then start a whole system backup, the ON-Bar process attempts to save the logical logs. Because the continuous logical log backup is running, an error message is returned indicating

that a logical log backup is already running, and the whole system backup returns a non-zero error code. In this case the logical logs are backed up only one time. To avoid the error, create a physical backup with the **onbar -b -w -p** command.

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. Alternatively, you can perform table level restores with the **archecker** utility.

Example: Back up a whole system

The following command performs a level-0 whole system backup after taking a checkpoint of all online storage spaces and logical logs:

```
onbar -b -w
```

The following command performs a level-1 whole system backup:

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

Example: Back up all online storage spaces and logical logs

The following command performs a standard, level-0 backup of all online storage spaces and used logical logs:

```
onbar -b
```

Example: Perform an incremental backup

The following command performs a standard, level-1 backup:

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

Example: Back up specified storage spaces and all logical logs

The following command performs a level-0 backup of the dbspaces named **fin_dbspace1** and **fin_dbspace2** and all logical logs:

```
onbar -b fin_dbspace1 fin_dbspace2
```

Example: Back up a list of storage spaces specified in a file

The following sample file named `listfile3` contains a list of storage spaces to be backed up:**blobsp2.1**, **my_dbspace1**, **blobsp2.2**, **dbsl.1**, **rootdbs.1**, and **dbsl.2**.

```
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
```

The following command backs up the storage spaces listed in the `listfile3` file:

```
onbar -b -f listfile3
```

Example: Back up logical logs

The following command starts a manual logical-log backup:

```
onbar -b -l
```

The following command backs up the current logical-log file:

```
onbar -b -l -c
```

Example: Physical backup

The following command backs up all storage spaces without backing up any logical logs:

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

A warning message is written to the ON-Bar activity log file stating that log file backup was not initiated. The message also contains the log unique ID of the latest log file that is required for a restore of the storage spaces. The latest required log file contains the archive checkpoint of the last dbspace backed up.

Example message:

```
2011-12-14 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.
```

Related tasks:

"Configuring a continuous log restore by using ON-Bar" on page 6-10

"Replacing disks during a restore" on page 6-14

"Preparing to back up data" on page 5-1

Related reference:

"Plan a backup system for a production database server" on page 2-3

"ON-Bar security" on page 4-9

List of storage spaces in a file

You can list storage spaces to back up or restore in a file.

The *filename* value can be any valid UNIX or Windows file name:

- Simple file names, for example: `listfile_1`)
- Relative file names, for example: `../backup_lists/listfile_2` or `..\backup_lists\listfile2`
- absolute file names, for example: `/usr//backup_lists/listfile3` or `c:\\backup_lists\listfile3`

The format rules for the file are:

- If you are restoring chunks, list storage space names without paths. Each line can list more than one storage space, separated by spaces or a tab.
- If you are renaming chunks, 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.
- Comments begin with a # or a ; symbol and continue to the end of the current line.
- ON-Bar ignores all comment or blank lines in the file.

Backing up blobspaces

You can back up blobspaces in a database that uses transaction logging.

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. You can run the **onstat -l** command to verify the logical-log status.

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.

Important: 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.

To back up blobspaces:

1. Verify the logical-log status by running the **onstat -l** or **xctl onstat -l** command.
2. Switch to the next log file by running the **onmode -l** command.
3. Back up the logical logs:
   - If the blobspace is online, run the **onbar -b -l -c** command.
   - If the blobspace is offline, run the **onbar -b -l -O** or **onbar -b -O** command. If this backup is successful, ON-Bar returns 178.
4. Back up the blobspaces by running the **onbar -b** or **onbar -b -w** command.

Related reference:

onstat -L command: Print the number of free locks (Administrator's Reference)

onbar -m syntax: Monitoring recent ON-Bar activity

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

Monitor recent ON-Bar activity

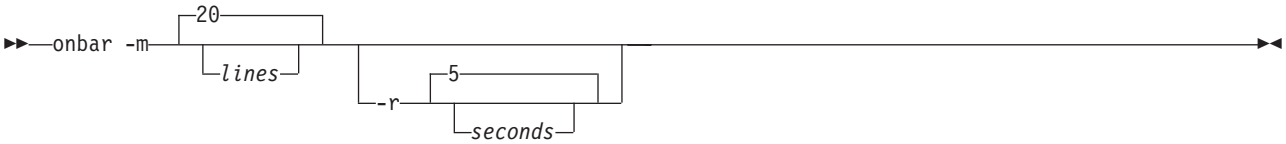


Table 5-2. Options for the onbar -m command

Option	Description
-m	Prints the recent activity of ON-Bar from the activity log file.
lines	Specifies the number of lines to output. Default is 20 lines.
-r	Causes the **onbar -m** command to repeat.
seconds	Specifies the number of seconds to wait before repeating. Default is 5 seconds.

Related concepts:

"bar_act.log file: ON-Bar activity log" on page 3-4

Related reference:

Chapter 10, "ON-Bar messages and return codes," on page 10-1

"Message format in the ON-Bar message log" on page 10-1

"ON-Bar security" on page 4-9

Viewing a list of registered backups

You can create a list of the registered ON-Bar backups performed on your system.

To view the list of registered backups:

1. Create a view in the **sysutils** database that contains information from the **bar_action**, **bar_instance**, and **bar_object** catalog tables. Include the following fields in the view:
   - Backup_ID: The internally generated ID for the backup
   - Type: Defines whether the backup is a whole system backup, dbspace backup, or logical log backup.
   - Object_Name: The name of the object backed up.
   - Ifx_Time: Time at which the object was created. For dbspace backups, the checkpoint time that started the backup. For logical logs, the time when the log become full.
   - CopyID_HI: The High part of the ID to locate the object in the storage manager.
   - CopyID_LO: The Low part of the ID to locate the object in the storage manager.
   - Backup_Start: Date and time when the backup started for this object
   - Backup_End: Date and time when the backup ended for this object.
   - Verify_Date: The time of the last verification made to this object, if any.
2. Run a SELECT statement against the view.

Example

The following statement creates a view that contains backup information:

```
CREATE VIEW list_backups(Backup_ID, Type, Object_Name, Ifx_Time, CopyID_HI,
            CopyID_LO, Backup_Start, Backup_End, Verify_Date)
AS SELECT * FROM (
SELECT
        act_aid AS backup_id,
        DECODE(act_type, 5, "Whole-System", DECODE(obj_type, "L",
           "Logical log", "Dbspace")) AS Type,
        substr(obj_name,1, 8) AS Object_Name,
        min(DBINFO ('utc_to_datetime', seal_time)) AS Ifx_Time,
        ins_copyid_hi AS CopyID_HI,
        ins_copyid_lo AS CopyID_LO,
        act_start AS Backup_Start,
        act_end AS Backup_End,
        ins_verify_date AS Verify_Date

FROM
        bar_action A,
        bar_instance I,
        bar_object O
WHERE
        A.act_aid = I.ins_aid AND
        A.act_oid = O.obj_oid AND
        A.act_oid = I.ins_oid AND
        O.obj_type in ("R", "CD", "ND", "L")
GROUP BY 1,2,3,5,6,7,8,9
ORDER BY Ifx_Time, Backup_ID) AS view_list_backups
```

The following query returns all the backups:

```
SELECT * FROM list_backups
```

Related reference:

The sysutils Tables (Administrator's Reference)

onbar -P syntax: Printing backed-up logical logs

You can use the **onbar -P** command to print logical logs that are backed up using the ON-Bar utility.

To run ON-Bar commands, you must be user **root**, user **informix**, a member of the **bargroup** group on UNIX, or a member of the Informix-Admin group on Windows.

- "Usage" on page 5-11
- "Example: Print a specific transaction" on page 5-11
- "Example: Print multiple logical log files" on page 5-12

Print backed-up logical logs

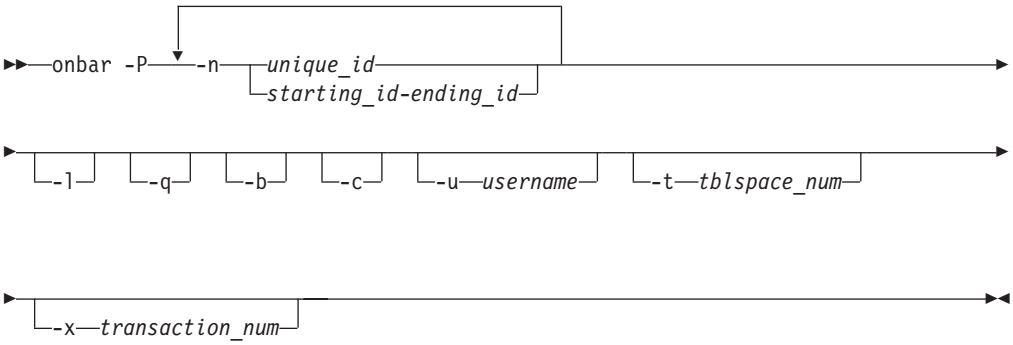


Table 5-3. Options for the onbar -P command

Option	Purpose
-b	Print logical-log records associated with blobspace blobpages. The database server stores these records on the logical-log backup media as part of blobspace logging.
-c	Use the compression dictionary to expand compressed data.
-l	Print the long listing of the logical-log record. The long listing of a log record includes complex hexadecimal and ASCII dumps of the entire log record.
-n *starting_id-ending_id*	Print the logical-log records contained in the specified range of log files. The *starting_id* option is the ID of the first log to print. The *ending_id* option is ID of the last log to print. The value of the *starting_id* option must be smaller than the value of the *ending_id* option. Separate the starting and ending ID values with a hyphen. Do not include blank spaces.
-n *unique_id*	Print the logical-log records contained in the specified log file. The *unique_id* option 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

Table 5-3. Options for the onbar -P command (continued)

Option	Purpose
-q	Do not print the program header
-t *tblspace_num*	Print the records associated with the tblspace that you specify with the *tblspace_num* option. Specify the *tblspace_num* value as either an unsigned integer or hexadecimal value. If you do not use a prefix of `0x`, the value is interpreted as an integer. The integer must be greater than zero and must exist in the **partnum** column of the **systables** system catalog table.
-u *username*	Print the records for a specific user. The user name must be an existing login name and conform to operating-system-specific rules for login names.
-x *transaction_num*	Print only the records associated with the transaction that you specify. 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 roll-forward. 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.

Usage

To view the backed-up logical logs, the storage manager must be running.

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

Example: Print a specific transaction

The following command prints information about a single transaction that was performed by the user **informix** against the tblspace 1048722 and is contained in the logical log file 2:

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

The output for this command might be:

```
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.
```

Example: Print multiple logical log files

The following command prints the logical log records for the logical logs files that have IDs of 2, 3, 4, 5, 10, 11, and 12:

```
onbar -P -n 2-5 -n 10-12
```

Related reference:

onstat -l command: Print physical and logical log information (Administrator's Reference)

onstat -L command: Print the number of free locks (Administrator's Reference)

SYSTABLES (SQL Reference)

"ON-Bar security" on page 4-9

onbar -v syntax: Verifying backups

Use the **onbar -v** command to verify that backups that were created by the ON-Bar utility are complete and can be restored.

To run ON-Bar commands, you must be user **root**, user **informix**, a member of the **bargroup** group on UNIX, or a member of the Informix-Admin group on Windows.

Sufficient temporary space must be available. For more information, see "Temporary space for backup verification" on page 5-14.

Verify backups

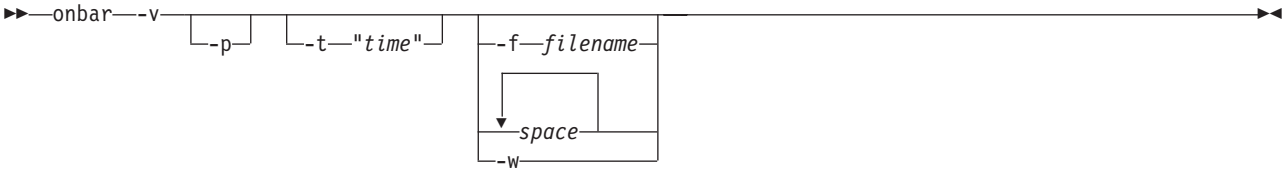


Table 5-4. Options for the onbar -v command

Option	Description
-v	Verifies a backup. The server can be in any mode. If verification is successful, you can restore the storage spaces safely. You can verify a whole-system or physical-only backup. You cannot verify the logical logs.
space	Names of storage spaces to verify. If you enter more than one storage-space name, use a space to separate the names.

Table 5-4. Options for the onbar -v command (continued)

Option	Description
-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 "List of storage spaces in a file" on page 5-7. The file can list multiple storage spaces per line.
-p	Verifies a physical-only backup.
-t "*time*"	Specifies the date and time to which dbspaces are verified. Must be surrounded by quotation marks. 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.

Usage

The **onbar -v** command runs the **archecker** utility. 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.

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`. IBM Software Support uses the `ac_msg.log` to diagnose problems with backups and restores.

The **onbar -v** command verifies only the smart-large-object extents in an sbspace. For a complete check, use the **oncheck -cS** command.

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.

Example: Perform a point-in-time verification of a backup

The following command verifies a backup at a point-in-time:

```
onbar -v -t "2011-12-10 10:20:50"
```

Example: Verify backups of storage spaces listed in a file

The following command verifies the backed-up storage spaces that are listed in the file `bkup1`:

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

Example: ON-Bar activity log verification messages

The following examples show messages about verification 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.
```

Example: archecker message log verification messages

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
```

Related tasks:

"Verifying an expired backup" on page 5-16

Related reference:

"ON-Bar security" on page 4-9

Temporary space for backup verification

When you verify backups, 15-25 MB of temporary space must be available.

During backup verification, the **archecker** utility requires about 15 MB of temporary space for a medium-size system (40-50 GB) and 25 MB 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 IBM Software Support can review them.

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

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

For IBM Informix, 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))
```

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

pagesize
The system page size in KB.

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.)

For example, you would need 12.9 megabytes of temporary disk space on a 50-gigabyte system with a page size of 2 KB. 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 KB to MB, divide the result by 1024:

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

Verification failures

The verification of a backup can fail for various reasons. If a backup fails verification, do not attempt to restore it.

The causes of a verification failure 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.

Backups with corrupted pages

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

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**.

Backups with corrupted control information

In this case, all the data is correct, but some of the backup control information is incorrect, which might cause problems with the restore. Ask IBM Software 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.

Diagnosing why a backup failed verification

If a backup failed verification, you can take steps to diagnose and attempt to fix the problem.

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 prints a message.
2. Back up the data onto different media.

   Do not reuse the original backup media because it might be damaged.

   Do not use any backups based on this backup. If the level-0 backup failed verification, do not use the corresponding level-1 and level-2 backups.
3. Verify this new backup. If verification succeeds, you can restore the storage spaces.
4. 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.
5. If verification fails again, call IBM Software 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 corrupted, IBM Software Support can help you determine which portion of the backup can be restored in an emergency.

   IBM Software Support might advise you to run **oncheck** options against a set of tables.

Verifying an expired backup

You can verify an expired backup in case subsequent backups are not valid.

To verify an expired backup:

1. Check that the status of the backup save set on the storage manager. If the storage manager 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.
3. Run the **onbar -v** command again.

Related reference:

"onbar -v syntax: Verifying backups" on page 5-12

Restoring when a backup is missing data

If a backup fails verification because of missing data, you can perform a restore from an older backup.

To restore when a backup is missing data:

1. Choose the date and time of an older backup than the one that failed. To perform a point-in-time verification, use the **onbar -v -t** *time space* command.
2. If the older backup passes verification, perform a point-in-time physical restore by using the same *time* value, then perform a log restore, as follows:

   ```
   onbar -r -p -t time space
   onbar -r -l
   ```

3. Expire the corrupted backup at your storage manager.
4. Run the **onsmsync** command 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, preventing ON-Bar from attempting to use such backups.

Related reference:

"The onsmsync utility" on page 8-4

"onbar -r syntax: Restoring data" on page 6-2

Chapter 6. Restore data with ON-Bar

You can use the ON-Bar utility to restore data that was backed up by the ON-Bar utility.

Before you restore data, use the pre-restore checklist to determine if whether a restore is needed and to prepare for a restore.

To perform a restore with the ON-Bar utility:

1. Make the storage devices that were available during the backup available for the restore.
2. If necessary, add enough temporary space to perform the restore. The logical log restore portion of a warm restore requires temporary space. The minimum amount of temporary space is equal to the smaller of the total amount of allocated logical-log space and the number of log files to be replayed.
3. Run the **onbar -r** command with the appropriate options to restore the data.
4. Monitor the ON-Bar activity log.
5. After the restore is complete, run the **onstat -d** command to verify that all storage spaces are restored. The letter O in the flags column indicates that the chunk is online.

Pre-restore checklist

Use this checklist to determine if a restore is necessary and to prepare for a restore.

To prepare for a restore:

- Determine if you need to restore. If one or more of these problems is true, you perform a restore 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?
- Diagnose the problem by using database server monitoring tools.
- If the root dbspace or the dbspaces that contain the physical log and logical-log files need to be restored, you must perform a cold restore. The database server must be offline during a cold restore. Ask your client users to log off the system.
- Contact the appropriate vendor to resolve the following types of problems before doing a restore:
  - The storage manager
  - The XBSA connection
  - The operating system
  - The storage media

Related reference:

 Database server monitoring (Administrator's Guide)

Storage space status and required actions

To determine the state of each storage space and its chunks, examine the output of the **onstat -d** command. The storage space status determines the action you need to take to solve the problem. The database server must be online.

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

Table 6-1. Chunk flag descriptions and required actions

chunk 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 is physically restored, but needs a logical restore.	Perform a logical restore.
L	Storage space is being logically restored.	Try the logical restore again.
N	Chunk is renamed and either down or inconsistent.	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.

Related reference:

onstat -d command: Print chunk information (Administrator's Reference)

"onbar -r syntax: Restoring data"

Storage device availability

Verify that the storage devices and files used in the backup are available for the restore.

If you drop a dbspace or mirror device after a level-0 backup, the dbspace or mirror device must be 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.

If you add a chunk after your last backup, the chunk device must be available to the database server when it rolls forward the logical log.

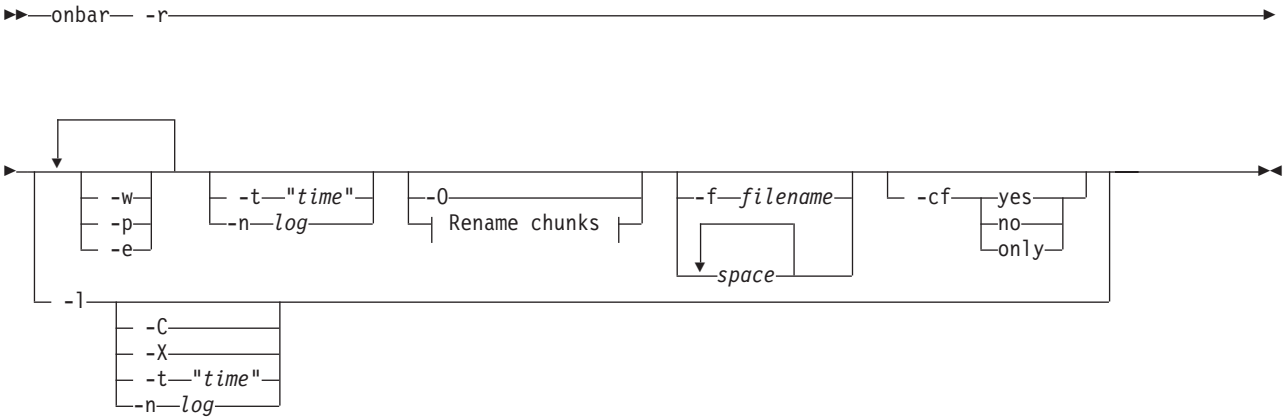
onbar -r syntax: Restoring data

To perform a complete restore, use **onbar -r** command.

To run ON-Bar commands, you must be user **root**, user **informix**, a member of the **bargroup** group on UNIX, or a member of the Informix-Admin group on Windows.

• "Usage" on page 6-6

Perform a restore



Rename chunks:

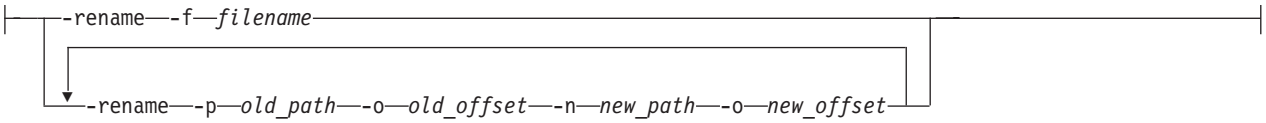


Table 6-2. Options for the onbar -r command.

Option	Description
-r	Specifies a restore. If the database server is offline, ON-Bar performs a cold restore. If the database server is in online, quiescent, or fast recovery mode, ON-Bar performs a warm restore. In a cold restore, the **-r** option restores all storage spaces and salvages and restores the logical logs. In a warm restore, the **-r** option restores all storage spaces that are offline and restores the logical logs. You must specify the **-r** option first.
space	Specifies which storage spaces to back up as a list of one or more dbspace, blobspace, or sbspace names, separated by blank spaces. ON-Bar restores only the storage spaces listed. If the database server is offline, you must list all the critical dbspaces. You cannot specify temporary spaces.

Table 6-2. Options for the onbar -r command. (continued)

Option	Description
-C	Continuously restores logical 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 affect continuous log restoration.
-cf	Specifies whether the critical files are restored during a cold restore. The critical files are the `onconfig` file, the `sqlhosts` file (on UNIX), the `oncfg_`*servername.servernum* file, and the `ixbar.`*servernum* file. Valid values are: • **yes** = Restores the critical files. • **no** = Default. Does not restore the critical files. • **only** = Restores only the critical files.
-e	Specifies an external restore. Run the **onbar -r -e** command after you externally restore the storage spaces. Marks storage spaces as physically restored, restores the logical logs, and brings the storage spaces online.
-f *filename*	Specifies the path and file name of a text file that lists the storages spaces to restore or rename. Use this option to avoid entering a long list of storage spaces. For more information, see "List of storage spaces in a file" on page 5-7.
-l	Specifies a logical restore only. Restores and rolls forward the logical logs. The logical restore applies only to those storage spaces that are already physically restored. **Important:** 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*.
-n *log*	Indicates the unique ID of the last logical log to be restored in a cold restore. The database server must be offline. To find the unique ID, use the **onstat -l** command. A point-in-log restore is a special 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 the specified log, ON-Bar does not restore them and their data is lost. If the specific logical log applies to more than one timeline, ON-Bar uses the latest one. Cannot be combined with the **-t** option.
-n *new_path*	Specifies the new path of the chunk. Use with the **-rename** option.

Table 6-2. Options for the onbar -r command. (continued)

Option	Description
-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, ON-Bar takes the storage space offline and then restores 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, ON-Bar recreates 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. 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. 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. Cannot be combined with the **-rename** option.
-o *new_offset*	Specifies the offset of the renamed chunk. Use with the **-rename** option.
-o *old_offset*	Specifies the offset of the chunk to be renamed. Use with the **-rename** option.
-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. If the LTAPEDEV configuration parameter is set to `/dev/null` or `NUL`, you must use the **-p** option during a restore.
-p *old_path*	Specifies the path of the chunk to be renamed. Use with the **-rename** option.
-rename	Renames one or more chunks during a cold restore. The database server must be offline. 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. You can rename chunks that have level-0 backups. Cannot be combined with the **-O** option.

Table 6-2. Options for the onbar -r command. (continued)

Option	Description
-t "*time*"	Specifies the time of the last transaction to be restored from the logical logs in a cold restore. The database server must be offline. All storage spaces specified are restored to the same point in time. However, if you perform a physical restore followed by a logical restore, the logical restore can be to a later point in time. For example you might detect that your current backup is corrupted, and that you need to restore the previous backup. In this case, start your physical restore with the timestamp of your previous backup, and subsequently start the logical recovery to a more recent timestamp. 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. To determine the appropriate date and time for the point-in-time restore, use the **onlog** utility. The **onlog** utility output shows the date and time of the committed transactions in the logical log. All data transactions that occurred after the time you specify in the restore command are lost. Use quotation marks 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. Cannot be combined with the **-n** *log* option.
-w	Performs a whole-system restore of all storage spaces and logical logs from the last whole-system backup. The database server must be offline. After the whole-system restore is complete, the server is in quiescent mode. If you specify **onbar -r -w** without a whole-system backup, return code 147 is returned because ON-Bar cannot find any storage spaces backed up as part of a whole-system backup.
-X	Stops continuous logical log restore. Leaves the server in quiescent mode in a logical restore suspend state without restoring additional logs.

Usage

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 chunks simultaneously during the restore, except for an external restore. You cannot specify to restore temporary spaces. When you restore the critical dbspaces (for example, the root dbspace), the database server recreates the temporary dbspaces, but they are empty.

ON-Bar restores the storage spaces in parallel if the BAR_MAX_BACKUP configuration parameter is set to a value greater than 1. To speed up restores, you can add additional CPU virtual processors.

You can restore noncritical storage spaces in a warm restore, when the database server is online, 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.

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 or allow ON-Bar to restore all down storage spaces by not explicitly specifying any spaces.

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.

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.

By default, ON-Bar restores the latest backup. If you do not want to restore the latest backup, you can restore from an older backup: for example, when backup verification failed or the backup media was lost. You can perform a point-in-time restore or a point-in-log restore. Alternatively, you can expire a bad backup in the storage manager, run the **onsmsync** command, and then restore from the older backup. If you accidentally drop a storage space, you can use a point-in-time restore or a point-in-log restore to recover it.

You can force a restore of online storage spaces (except critical dbspaces) by using the **-O** option. 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.

You can restore critical files during a cold restore by including the **-cf yes** option.

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.

Example: Perform a whole-system restore

A whole-system restore is a cold restore and must be performed while the server is offline. The following command restores a whole-system backup:

```
onbar -r -w
```

Example: Restore specific storage spaces

The following example restores two specific storage spaces, **fin_dbspace1** and **fin_dbspace2**:

```
onbar -r fin_dbspace1 fin_dbspace2
```

Example: Perform a warm restore in stages

The following commands perform a physical restore, back up logical logs, and perform a logical restore:

```
onbar -r -p
onbar -b -l
onbar -r -l
```

Example: Point-in-time restore

The following command restores database server data to its state at a specific date and time:

```
onbar -r -t "2011-05-10 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.

Example: Point-in-time restore with a French locale

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 following command restores the data to a specific point in time that is formatted for the French locale:

```
onbar -r -t "Lundi 6 Juin 2011 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. For example:

```
%.1d %B %iy %H:%M:%S
```

The following command restores to a specific point in time by specifying a two-digit year for the French locale:

```
onbar -r -t "6 Juin 11 11:20:14"
```

Example: Point-in-time restore in stages

The following commands perform a physical restore and a logical restore to the same point in time:

```
onbar -r -p -t "2011-05-10 11:35:57"
onbar -r -l -t "2011-05-10 11:35:57"
```

Example: Restore a dropped storage space and chunks

Suppose that a transaction dropped a storage space named **dbspace1** and deleted chunks at the time 2011-05-10 12:00:00. The following command restores the storage space and recreates the deleted chunks while the server is offline:

```
onbar -r -t "2011-05-10 11:59:59" -O
```

Example: Restore critical files

The following command restores data and the critical files during a cold restore:

```
onbar -r -cf yes
```

Related tasks:

"Restoring when a backup is missing data" on page 5-16

"Replacing disks during a restore" on page 6-14

Related reference:

"Storage space status and required actions" on page 6-2

"External restore commands" on page 7-6

"ON-Bar security" on page 4-9

"onbar -RESTART syntax: Restarting a failed restore" on page 6-18

Avoid salvaging logical logs

The **onbar -r** command automatically salvages the logical logs. However, avoid salvaging logical logs in some situations.

Use the **onbar -r -p** and **onbar -r -l** commands to skip log salvage.

If you set the LTAPEDEV configuration parameter to `/dev/null` on UNIX or to `NUL` on Windows, the logical logs are not salvaged in any ON-Bar restore (**onbar -r** or **onbar -r -w**, for example).

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.
- 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.

Related tasks:

"Restoring to a different computer" on page 6-16

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.

If you try to perform a cold restore without a backup, data in the storage spaces that were not backed up are lost.

To perform a cold restore:

1. Shut down the server by running the **onmode -ky** command.
2. If the disk that contains the logical-log files must be replaced or repaired, use the **onbar -b -l -s** command to salvage logical-log files on the damaged disk. Otherwise, ON-Bar automatically salvages the logical logs.
3. If necessary, repair or replace the damaged disk.
4. If the files in `INFORMIXDIR` are damaged, copy the back ups of administrative files to their original locations.

   Otherwise, you do not need to copy the administrative files.
5. Restore the critical and noncritical storage spaces by running the **onbar -r** command. When the restore is complete, the database server is in quiescent mode.
6. Start the server by running the **onmode -m** command.
7. Synchronize the storage manager by running the **onsmsync** command.

Related reference:

onmode -k, -m, -s, -u, -j: Change database server mode (Administrator's Reference)

"The onsmsync utility" on page 8-4

Configuring a continuous log restore by 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.

The version of IBM Informix must be identical on both the primary and secondary systems.

To configure continuous log restore by using ON-Bar:

1. On the primary system, perform a level-0 backup with the **onbar -b -L 0** command.
2. Import the backup objects that were created to the storage manager of the secondary server.
3. On the secondary system, perform a physical restore with the **onbar -r -p** command. 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 **onbar -b -l** command.
5. Transfer the backed up logical logs to the secondary system and restore them with the **onbar -r -l -C** command.
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, run the **onbar -r -l** command.
   - After all available logical logs are restored, run the **onbar -r -l -X** command.

Related concepts:

"Continuous log restore" on page 1-8

Related reference:

"onbar -b syntax: Backing up" on page 5-2

Restoring data by 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.

In a mixed restore, you first perform a cold restore of the critical dbspaces (the root dbspace and the dbspaces that contain the physical and logical logs) and the dbspaces containing your urgent data. Because you do not restore all dbspaces, you can bring the server online faster. You then restore the remaining storage spaces in one or more warm restores.

To perform a mixed restore:

1. Shut down the database server by running the **onmode -ky** command.
2. Perform a cold restore of the critical and urgent dbspaces by running the **onbar -r** command with the list of critical and urgent dbspace names. You can specify a point in time to restore from an older backup.
3. Start the server by running the **onmode -m** command.
4. Synchronize the storage manager by running the **onsmsync** command.

5. Perform a warm restore of the remaining storage spaces by running the **onbar -r** command. You can perform multiple warm restores to prioritize certain storage spaces.

Examples

Example 1: Simple mixed restore

A database server has five dbspaces in addition to the root dbspace: **logdbs**, **dbs_1**, **dbs_2**, **dbs_3**, and **dbs_4**. 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**. The dbspace that contains urgent data is **dbs_1**. The following commands shut down the database server, perform a cold restore on the critical and urgent dbspaces, and restart the database server:

```
onmode -ky
onbar -r rootdbs logdbs dbs_1
onmode -m
```

After the database server starts, any data stored in **rootdbs**, **logdbs**, and **dbs_1** dbspaces is accessible.

The following commands synchronize the storage manager and perform a warm restore of the remaining dbspaces, **dbs_2**, **dbs_3**, and **dbs_4**:

```
onsmsync
onbar -r
```

Example 2: Point-in-time mixed restore

The following commands perform a cold restore for a subset of the storage spaces (including all critical dbspaces) in the initial cold restore, perform a warm restore for **dbspace_2** and **dbspace_3**, followed by a warm restore of **dbspace_4** and **dbspace_5**, and finally perform a warm restore of all remaining storage spaces:

```
onbar -r -t "2011-05-10 11:35:57" rootdbs logspace_1 dbspace_1
onmode -m
onsmsync
onbar -r dbspace_2 dbspace_3
onbar -r dbspace_4 dbspace_5
onbar -r
```

Strategies for using a mixed restore

To implement a mixed-restore strategy, carefully select the set of dbspaces in which you place your databases and database objects when 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 restore the dbspaces containing the database catalogs, table data, and index: in this case, the dbspaces **cat_dbs**, **tab_dbs_1**, **tab_dbs_2**, and **idx_dbs**.

To simplify the management and tracking of your data, 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: 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: 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: 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 that 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
```

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
```

In a larger system with dozens of dbspaces, you can 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.

Recreating chunk files during a restore

If the disk or file system fails, one or more chunk files might be missing from the dbspace. Use the `-0` option to recreate missing chunk files and any necessary directories during a restore.

The restore fails if insufficient 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.

Restoring when using cooked chunks

You can recreate missing cooked chunk files during a restore.

Restriction: ON-Bar does not recreate chunk files during a logical restore if the logical logs contain chunk-creation records.

To restore when using cooked chunks:

1. Install the new disk.
2. Based on your system, perform one of the following tasks:
   - On UNIX, mount the device as a file system.
   - On Windows, format the disk.
3. Allocate disk space for the chunk file.
4. Run the **onbar -r -O** *space* command to recreate the chunk files and restore the dbspace.

Related tasks:

Allocating cooked file spaces on UNIX (Administrator's Guide)

Restoring when using raw chunks

You can recreate missing raw chunk files during a restore.

To restore when using raw chunks:

1. Install the new disk.
2. For UNIX, 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.
3. Issue the **onbar -r** *space* command to restore the dbspace.

Related tasks:

Allocating raw disk space on UNIX (Administrator's Guide)

Allocating raw disk space on Windows (Administrator's Guide)

Reinitializing the database server and restoring data

Reinitializing disk space destroys all existing data managed by the database server. However, you can restore data from a backup that was performed before reinitialization.

You must have a current, level-0 backup of all storage spaces.

During initialization, ON-Bar saves the emergency boot file elsewhere and starts a new, empty emergency boot file. Therefore, any backups that you performed before reinitializing the database server are not recognized. You must use the copy of the emergency boot file you saved before initialization to restore the previous database server instance.

To reinitialize the database server and restore the old data:

1. Copy the emergency boot file, the `oncfg` file, and the `onconfig` file to a different directory.
2. Set the FULL_DISK_INIT configuration parameter to 1 in the `onconfig` file.
3. Shut down the database server.
4. Reinitialize the database server by running the **oninit -i** command.
5. Move the administrative files into the database server directory. If the administrative files are unavailable, copy them from the last backup into the database server directory.
6. Perform a restore by running the **onbar -r -p -w** command. Do not salvage the logical logs.

7. Verify that you restored the correct instance of the critical and noncritical storage spaces.

Related reference:

The oninit utility (Administrator's Reference)

Replacing disks during a restore

You can replace disks during a restore by renaming chunks. You 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.

The old chunk must be included in the last level-0 backup.

The following guidelines apply to 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 an `N` flag in the output of the **onstat -d** command.
- The new chunk must have the correct permissions.
- The new chunk must be included in the last level-0 backup.
- The new chunk path name and offset cannot overlap existing chunks.

Tip: If you use symbolic links to chunk names, you might not need to rename chunks; you only need to edit the symbolic name definitions.

To rename chunks during a restore:

1. Shut down the database server.
2. Run the **onbar -r** command with the **-rename** option and the chunk information options. If you are renaming the primary root or mirror root chunk, ON-Bar updates the values of the ROOTPATH and ROOTOFFSET, or MIRRORPATH, and MIRROROFFSET configuration parameters. The old version of the `onconfig` file is saved as `$ONCONFIG.`*`localtime`*.
3. Perform a level-0 archive so that you can restore the renamed chunks.

Examples

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

Example 1: Rename chunks by supplying chunk information in the command

The following command renames the chunks **chunk1** to **chunk1N** and **chunk2** to **chunk2N**:

```
onbar -r -rename -p /chunk1 -o 0 -n /chunk1N -o 20000
         -rename -p /chunk2 -o 10000 -n /chunk2N -o 0
```

Example 2: Rename chunks by supplying chunk information in a file

Suppose that you have a file named `listfile` that has the following contents:

```
/chunk1 0 /chunk1N 20000
/chunk2 10000 /chunk2N 0
```

The following command renames the chunks **chunk1** to **chunk1N** and **chunk2** to **chunk2N**:

```
onbar -r -rename -f listfile
```

Related reference:

"onbar -r syntax: Restoring data" on page 6-2

"onbar -b syntax: Backing up" on page 5-2

Renaming a chunk to a nonexistent device

To rename a chunk to a nonexistent device, specify the new path name, but restore the storage spaces 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 with the following command: **onbar -r -rename -p /chunk3 -o 0 -n /chunk3N -o 0**
2. When you see the following prompt, 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 is not yet restored to `/chunk3N`.
3. Perform a level-0 archive.
4. Add the physical device for `/chunk3N`.
5. Perform a warm restore of **sbspace1** with the **onbar -r sbspace1** command.
6. Perform a level-0 archive.

Restoring to a different computer

You can back up data on one computer and restore the data on a different computer. Importing a restore is useful for disaster recovery or upgrading a database server. After you back up your data and move over the storage-manager objects, you can perform an imported restore. An imported restore involves copying files from the source to the target computer and performing the restore in one of several ways.

Prerequisites:

- Your storage manager must support imported restores.
- A whole-system backup must include all storage spaces; logical logs are optional.

  The level-0 backup must include all storage spaces and logical logs.
- Both the source and target computers must be on the same LAN or WAN and must have the following attributes:
  - Identical hardware and operating systems
  - Identical database server versions
  - The same configuration and ROOTPATH information, although the server names and numbers can differ.
  - Identical storage-manager versions
  - Compatible XBSA libraries

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.

To perform the imported restore:

1. Install the database server and the storage manager on the target computer.
2. Set up the storage manager on the target database server instance.
   a. Set the necessary environment variables.
   b. Define the same type of storage devices as on the source instance.
   c. Label the storage media with the correct pool names.
   d. Mount the storage devices.
   e. Update the `sm_versions` file on the target computer with the storage-manager version.
3. Be sure that the target computer has the devices and links in place for the chunks that match the devices and links on the source computer
4. Perform a level-0 backup (**onbar -b** or **onbar -b -w**) of all storage spaces on the source database server.

   Restriction: Do not perform an incremental backup.
5. 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 tapes, 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 8-mm tape or disk.
   - If the backup is on the backup server, retrieve the backup from that backup server.

6. Use storage-manager commands to add the source host name as a client on the target computer.
7. Copy the following files from the source computer to the target computer.

Table 6-3. Administrative files to copy

File	Action
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. The file name is `ixbar.`*`servernum`*.
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 file name must match the DBSERVERNAME and SERVERNUM on the target computer.
The `onconfig` file	In the `onconfig` file, update the DBSERVERNAME and SERVERNUM parameters with the target database server name and number.
Storage-manager configuration files, if any	The storage-manager configuration files might need updating.

8. Restore the data in one of the following ways:

Table 6-4. Restore data options

Option	Action
If you did not start an Informix instance on the target server	Use the **onbar -r** command to restore the data.
If you are importing a whole-system backup	Use the **onbar -r -w -p** command to restore the data.
If you started an Informix instance on the target server.	Restore the data in stages: 1. Use the **onbar -r -p** command to restore the physical data. 2. Use the **onbar -r -l** command to restore the logical logs. This process avoids salvaging the logs and any potential corruption of the instance.

9. Before you expire objects on the target computer and the storage manager with the **onsmsync** utility, perform one of the following tasks. Otherwise, **onsmsync** expires the incorrect objects.
   - Manually edit the emergency boot file `viz ixbar.`*`servernum`* in the `$INFORMIXDIR/etc` directory on the target computer. Replace the IBM Informix server name that is used on the source computer with the IBM Informix server name of the target computer

- Run the **onsmsync -b** command as user **informix** on the target computer to regenerate the emergency boot file from the **sysutils** database only. The regenerated emergency boot file reflects the server name of the target computer.

Related reference:

"Avoid salvaging logical logs" on page 6-9

onbar -RESTART syntax: Restarting a failed restore

If a failure occurs with the database server, media, storage manager, or ON-Bar during a restore, you can restart the restore from the place that it failed. To restart a failed restore, the RESTARTABLE_RESTORE configuration parameter must be set to ON in the `onconfig` file when the restore fails.

Restart a restore

Table 6-5. onbar -RESTART command

Option	Description
-RESTART	Restarts a restore after a database server, storage manager, or ON-Bar failure. The RESTARTABLE_RESTORE configuration parameter must be set to ON when the restore failure occurs. You can restart the following types of restores: • Whole system • Point in time • Storage spaces • Logical part of a cold restore Do not use the **-RESTART** option if a failure occurs during a warm logical restore.

Usage

When you enable restartable restore, the logical restore is slower if many logical logs are restored. However, you save time if the restore fails and you restart the restore. Whether a restore is restartable does not affect the speed of the physical 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, all chunks of that storage space are restored. If storage spaces and incremental backups are restored successfully before the failure, they are not restored again.

If the BAR_RETRY configuration parameter is set to 2, ON-Bar automatically tries to restore any failed storage spaces and logical logs again. If the restore is successful, you do not need to restart the restore.

If the BAR_RETRY configuration parameter is set to 0 or 1, ON-Bar does not try to restore any failed storage spaces and logical logs again. If the database server is

still running, ON-Bar skips the failed storage space and attempts to restore the remaining storage spaces. To complete the restore, run the **onbar -RESTART** command.

The following figure shows how a restartable restore works when the restore failed during a physical restore of **dbspace2**. 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.

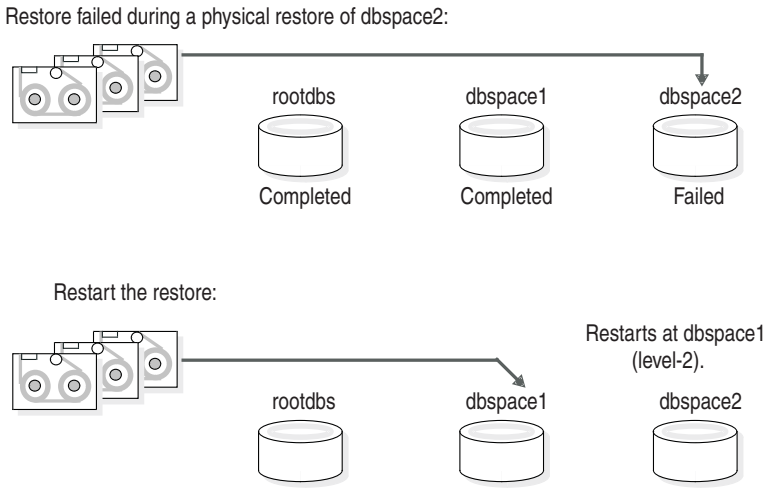


Figure 6-1. Restartable physical restore

If a restore fails during the logical phase and you restart the restore, ON-Bar verifies that the storage spaces are restored, skips the physical restore, and restarts the logical restore. The following figure 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 the restore at the place of failure.

Important: If a failure occurs during a warm logical restore, restart the restore from the beginning. If the database server is still running, run the **onbar -r -l** command to complete the restore.

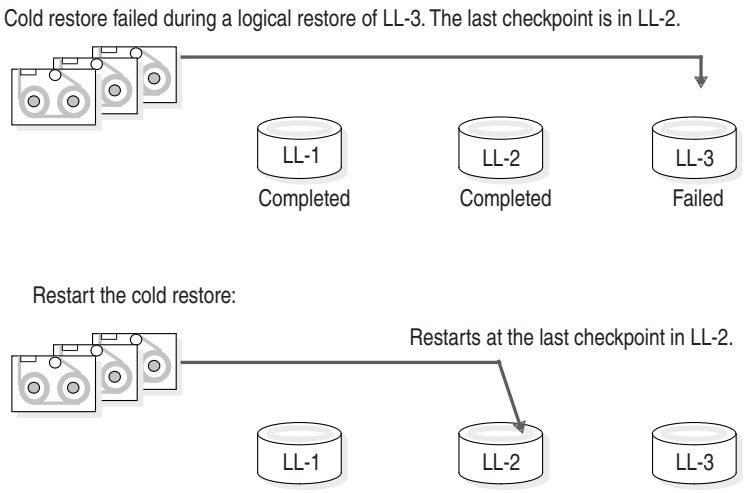


Figure 6-2. Restartable cold logical restore

Related reference:

"onbar -r syntax: Restoring data" on page 6-2
"BAR_RETRY configuration parameter" on page 17-11
"RESTARTABLE_RESTORE configuration parameter" on page 17-19
"ON-Bar security" on page 4-9

Resolve a failed restore

How you resolve a failed restore depends on the cause of the failure.

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.

The following table shows what results to expect when physical restore fails and the value of the BAR_RETRY configuration parameter is > 1.

Table 6-6. 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 tries each failed restore again. If the storage manager failed, fix the storage-manager error. If the tried 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.

Table 6-6. Failed physical restore scenarios (continued)

Type of error	RESTARTABLE_ RESTORE setting	What to do when the physical restore fails?
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.

The following table shows what results to expect when logical restore fails.

Table 6-7. 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 restarts at the failed logical log. If **onbar -r -l** still fails, shut down and restart the database server. The database server completes a fast recovery. All logical logs that were not restored are lost. If fast recovery does not work, you must 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	ON or OFF	ON-Bar tries each failed logical restore again. If the tried restore fails, the logical restore is suspended. Fix the storage-manager error. Then issue the **onbar -r -l** command. The restore restarts at the failed logical log.

Related reference:

"BAR_RETRY configuration parameter" on page 17-11

"RESTARTABLE_RESTORE configuration parameter" on page 17-19

Chapter 7. External backup and restore

These topics discuss recovering data by 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 IBM 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 vendor 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 7-5.

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.

The following figure shows how to perform a backup with mirroring.

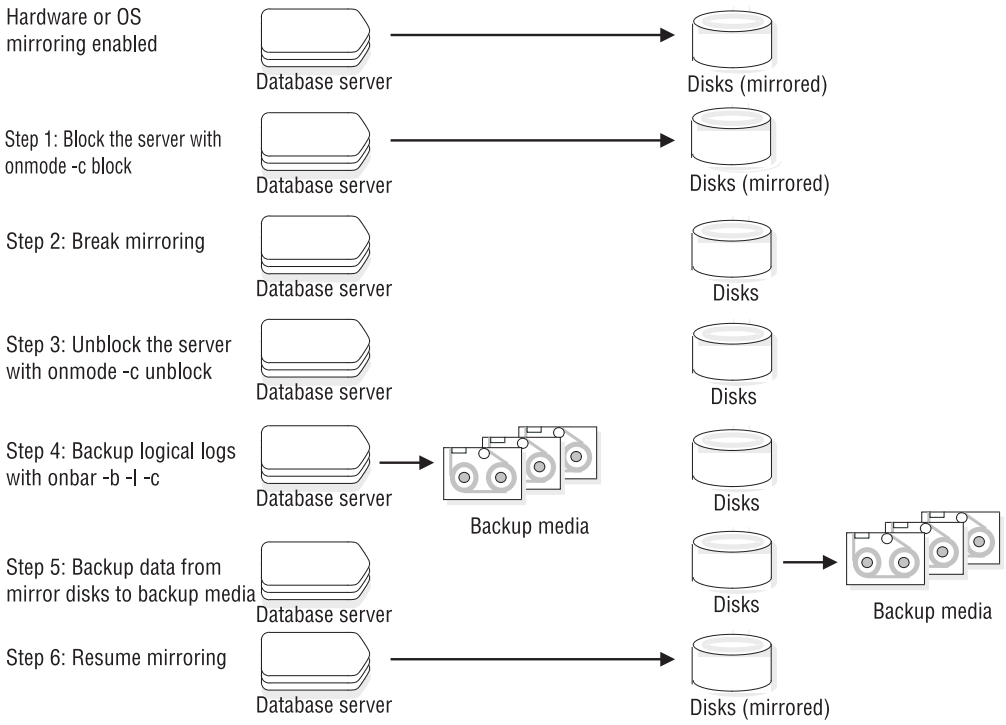


Figure 7-1. Perform a backup with mirroring

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 back up media with external commands. Now you can resume mirroring.

Block before backing up

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 by 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, you should 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 when chunks are not mirrored" on page 7-4.

Rules for an external backup

Before you begin an external backup, review the rules for performing an external backup.

The rules that you must follow are:

- 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.

  To stop continuous logical-log backup, use the **CTRL-C** command. To resume continuous logical-log backup, use the **onbar -b -l -C** command.
- 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.

- On AIX® operating systems, if the server is running with concurrent I/O because the DIRECT_IO configuration parameter is set to enable concurrent I/O, an online external backup program must also use concurrent I/O.

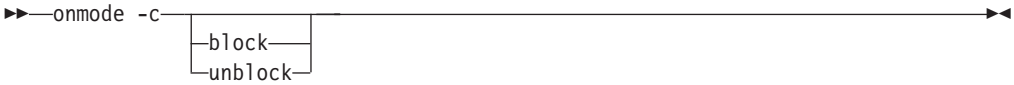
Important: Because the external backup is outside the control of ON-Bar, you must track these backups manually. For more information, see "Track an external backup."

Prepare for an external backup

These topics describe the commands used to prepare for an external backup. For the procedure, see "Performing an external backup when chunks are not mirrored" on page 7-4.

Block and unblock database server

This topic shows the syntax of the block and unblock commands on IBM Informix.



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**

Track 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.

The following table shows which items we recommend that you track in an external backup. ON-Bar keeps a limited history of external restores.

Table 7-1. 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 when chunks are not mirrored

The database server must be online or in quiescent mode during an external backup.

To perform an external backup when chunks are not mirrored:

1. To obtain an external backup, block the database server with the **onmode -c block** command. The system takes a checkpoint and suspends all update transactions. Users can access the database server in read-only mode.
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 with the **onmode -c unblock** command.
4. Back up all the logical logs including the current log so that checkpoint information is available for the external restore.

   Important: 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 **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 with the **onbar -b -l -c** command.

If you lose a disk, or the whole system, you are now ready to perform an external restore.

RS secondary server external backup

You can perform an external backup of an RS secondary server. Performing a backup of an RS secondary server blocks that RS secondary server, but does not block the primary server.

You can perform a logical restore from the logs backed up from the primary instance. The backup obtained from the secondary server cannot be restored with level-1 or level-2 backups.

Important: The external backup is not completed if the database instance contains any of the following:

- Nonlogging smart large objects
- Regular blobspaces
- Nonlogging databases
- Raw tables

If an external backup is performed on an instance that contains any of the previously mentioned items, then the backup is incomplete and cannot be used to restore the primary server.

If the backup fails because the checkpoint from the primary has timed out, you can use the BAR_CKPTSEC_TIMEOUT configuration parameter to increase the amount of time, in seconds, that an RS secondary server should wait for a checkpoint to arrive from the primary server while performing an external backup.

Performing an external backup of an RS secondary server

To perform an external backup of an RS secondary server, the STOP_APPLY configuration parameter must not be enabled. If STOP_APPLY is enabled, an error is returned. The server switches to STOP_APPLY mode when a backup is performed on an RS secondary. After the archive checkpoint is processed, the RS secondary server stops applying logical logs, but continues receiving logs from the primary server.

To perform an external backup of an RS secondary server that has a DELAY_APPLY configuration parameter value greater than 0, it might be necessary to temporarily decrease the parameter value. Performing the backup requires that the RSS process a checkpoint in the logical log, and if no checkpoint is observed within the amount of time that is specified by the **onmode -c block** *timeout* command in the second step of the following procedure, a backup is not permitted. The DELAY_APPLY configuration parameter can be decreased by the **onmode -wf DELAY_APPLY=***setting* command.

The primary database server must be online or in quiescent mode during an external backup.

To perform an external backup:

1. Ensure that the LOG_STAGING_DIR configuration parameter on the RS secondary server is set to point to a valid staging directory.
2. To obtain an external backup, block the database server with the **onmode -c block** *timeout* command. The *timeout* parameter indicates the number of seconds that the RS secondary server waits to receive a checkpoint. The *timeout* parameter is valid only when the **onmode -c block** command is run on an RS secondary server. You must wait for the **onmode -c block** command to return successfully before you proceed with the external backup.
3. 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.
4. To resume normal operations, unblock the database server by using the **onmode -c unblock** command.
5. After you perform the external backup, back up the current log and any new logs with the ON-Bar or **ontape** utilities.

   Important: Logical log backup is only possible on the primary server.

   If the DELAY_APPLY configuration parameter is set, the logs that are required for the restore process are not necessarily those logs that are currently active on the primary server because some logs could already be archived.

After the backup completes, if the DELAY_APPLY setting on the RS secondary server was decreased, it can be set to its original value by the **onmode -wf DELAY_APPLY=***setting* command. After an external backup, you can perform an external restore if a disk or the whole system fails.

Data 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 **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.

Restriction: 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**).

Rename chunks

You can rename chunks in an external cold restore by 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
```

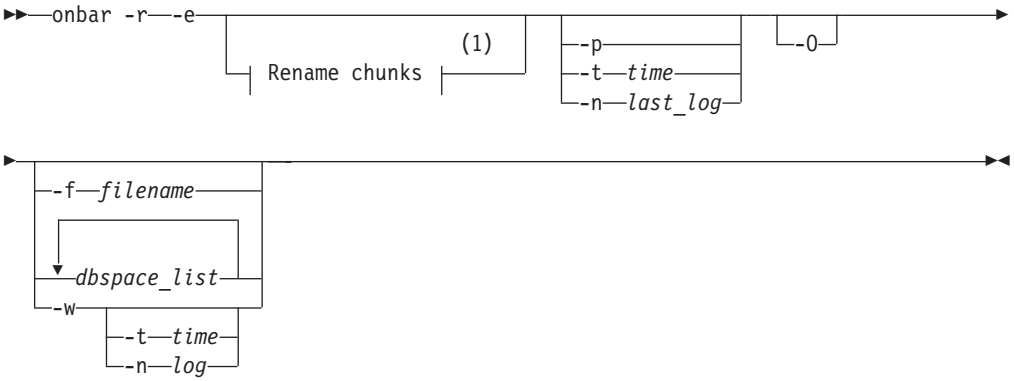
or

```
onbar -r -e rename -p old_path -o old_offset-n new_path-o new_offset
```

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 “onbar -b syntax: Backing up” on page 5-2

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.
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	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.
-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.
-w	Performs a whole-system restore of all storage spaces and logical logs from the last whole-system backup	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.

Related reference:

"onbar -r syntax: Restoring data" on page 6-2

Rules for an external restore

External restores have specific rules.

The following rules apply to external restores:

- You must externally restore from an external backup. Although the external backup is treated as a level-0 backup, it might actually be an incremental backup not related to IBM Informix.
- 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 with ON-Bar.
- If the external backups are from different times, the external restore uses the beginning logical log from the oldest backup.
- You cannot perform a mixed restore. If critical dbspaces must be restored, you must perform a full cold restore.

The following rules apply to external cold restores:

- 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.
- 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 be simultaneous. All critical dbspaces must have to be backed up within the same set of **onmode -c block ... onmode -c unblock** commands.

Performing an external restore

This section describes procedures for performing cold and warm external restores.

Performing a cold external restore

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 with the **onmode -ky** command.
2. Salvage the logical logs with the **onbar -b -l -s** command.
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 **onbar -r -e** command.
5. To perform a point-in-time external restore of all storage spaces, use the **onbar -r -e -t** *datetime* command.

   This step brings the database server to fast-recovery mode.

   ON-Bar and the database server roll forward the logical logs and bring the storage spaces online.

Performing a warm external restore

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.
   - To restore selected storage spaces and all logical logs, use the **onbar -r -e** *dbspace_list* command.
   - To restore the down noncritical storage space named dbsp1 and logical logs in separate steps, use the following commands:

     onbar -r -e -p dbsp1

     onbar -r -l dbsp1

- To restore all the noncritical storage spaces and logical logs, use the **onbar -r -e -O** command.

Examples of external restore commands

The following table contains examples of external restore commands.

External restore command	Action	Comments
onbar -r -e	Complete external restore	In a cold restore, restores everything. In a warm restore, restores all down noncritical storage spaces.
onbar -r -e -p **onbar -r -l**	Physical external restore and separate logical restore	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.
onbar -r -e *dbspace_list*	External restore of selected storage spaces and logical logs	Use this command in a warm external restore only.
onbar -r -e -p *dbspace_list* **onbar -r -l**	External restore of selected storage spaces and separate logical restore	Use this command in a warm external restore only.
onbar -r -e -t *datetime*	External point-in-time (cold) restore	Be sure to select a collection of backups from before the specified time.
onbar -r -e rename -p *old_path* **-o** *old_offset***-n** *new_path* **-o** *new_offset*	External (cold) restore with renamed chunks	Use this command to rename chunks in cold external restore only.
onbar -r -e -w **onbar -r -e -p -w**	Whole-system external restore	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

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 **onmode -c block** command.
2. Externally back up all chunks on the source database server.
3. When the backup completes, unblock the source database server with the **onmode -c unblock** command.
4. Make the source database server the primary server with the following command: **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 with the **onbar -r -e -p** command. On HDR, secondary server can restore only level-0 archives.
7. Make the target database server the secondary server with the following command: **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 with the **onbar -r -l** command.

   The `database server operational` messages appear in the message log on the primary and secondary servers.

Chapter 8. Customize and maintain ON-Bar

These topics discuss the following:

- Customizing ON-Bar and storage-manager commands with the **onbar** script
- Starting **onbar-worker** processes manually
- Expiring and synchronizing the backup history

Customizing ON-Bar and storage-manager commands

You can edit the script that is installed with ON-Bar to customize backup and restore commands, and storage manager commands.

On UNIX operating systems, the **onbar** shell script is in the `$INFORMIXDIR/bin` directory. On Windows operating systems, the **onbar.bat** batch script is in the `%INFORMIXDIR%\bin` directory.

Edit the script and backup a copy of the original file in case you need to revert to it.

Important: Edit the script with caution and test your changes. Do not change the cleanup code at the bottom of the script. Doing so might result in unexpected behavior, for example, leftover temporary files during backup verification.

The 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

  This section removes the **archecker** temporary files.
- End cleanup processing here

  Use this section to return **onbar_d** error codes.

Related reference:

Chapter 5, “Back up with ON-Bar,” on page 5-1

Updating the ON-Bar script during reinstallation

After you reinstall the database server, you might need to update the script that is installed with ON-Bar. Existing customized scripts were backed up by the installation process so that you can reuse the content.

The installation program installs the default **onbar** shell script on UNIX and the default **onbar.bat** batch script on Windows. If the default script differs from the local script, the installation program backs up the local script and issues a message to inform you that the local script was renamed. The naming convention of the renamed file is `onbar.`*`date`*, where *date* is the date when the file was renamed. For example, the file `onbar.2012.05.15` was renamed on May 15, 2012.

You can update the default script by adding information to it from the renamed script.

Print 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.

The following example is for UNIX:

```
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
```

The following example is for Windows:

```
@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
```

Migrate 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:

The following example is for UNIX:

```
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
```

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
```

Expire and synchronize 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, 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.

Choose 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 a specified period.

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.

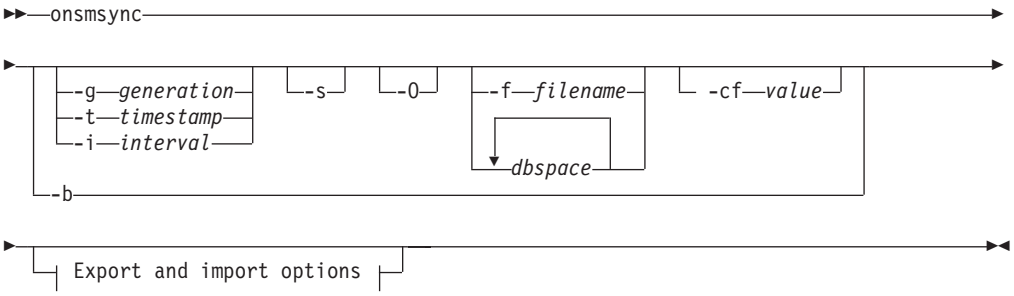
ON-Bar expires all level-1 backups older than the specified time unless they are required to restore from the oldest retained level-2 backup.

ON-Bar retains a whole-system backup that starts before the specified retention time and ends after the specified retention time.

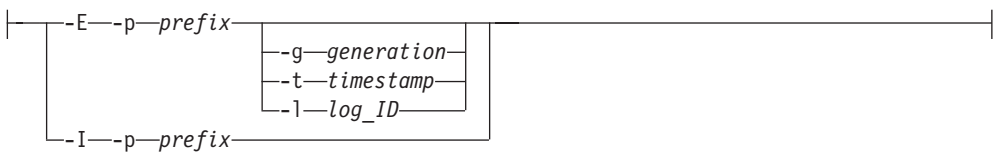
The onsmsync utility

Use the **onsmsync** utility to synchronize the **sysutils** database and emergency boot file with the storage manager catalog.

If your storage manager is the Informix Primary Storage Manager, you can also use the export and import options of the **onsmsync** utility to export Informix Primary Storage Manager backup objects to an external device and import objects from an external device to a device that is managed by the Informix Primary Storage Manager. You cannot use the export and import options with other storage managers.



Export and import options (to use only with the Primary Storage Manager):



The following table lists all **onsmsync** command elements, except the elements you use to import and export backup generations. The command elements that you use for importing and exporting are listed in Table 8-2 on page 8-6.

Table 8-1. Elements for **onsmsync** *commands*

Element	Purpose	Key considerations
-b	Regenerates 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** re-creates the `ixbar` file and populates it from the **sysutils** tables. If the `ixbar` file 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 was rebuilt, but is empty because it does not contain data, **onsmsync -b** recreates the **sysutils** data from the `ixbar` file. Do not use the **-b** element with the other **onsmsync** options. The **-b** element does not synchronize with the storage manager.
dbspace	Specifies the storage space or storage spaces to expire	If you enter the name of more than one storage space, use a blank 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 file name can be any valid UNIX or Windows file name.
-g *generation*	Specifies the number of versions of each level-0 backup to retain	The latest generation of backups is retained and all earlier ones are expired.
-i *interval*	Specifies the time interval for retaining backups.	The utility: • Retains backups that were created after this interval. • Expires backups that were created before this interval and removes the backups if expired objects are also removed. 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
-O	Overrides internal error checks and enforces expiration policy	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 "Expire all backups" on page 8-9.
-s	Skips synchronizing expired backups	The object expiration is based on other arguments if the **-s** option is provided.
-t *timestamp*	Expires all backups before a particular date and time	Retains backups that completed after the specified timestamp. Backups that occurred before the timestamp are not expired if they are needed to restore from other backups that occurred after the timestamp. Use the ANSI or GLS_DATETIME format for the timestamp.
-cf	Specifies whether to expire the critical files. When used with the **-g**, **-i**, or **-t**, deletes critical file backups from the Informix Primary Storage Manager	The critical files are the `onconfig` file, the `sqlhosts` file (on UNIX), the `oncfg_`*`servername.servernum`* file, and the `ixbar.`*`servernum`* file. Valid values are: • **yes** = Deletes backups of the critical files. • **no** = Does not delete backups of the critical files. • **only** = Deletes only the backups of critical files.

Table 8-2. Elements for **onsmsync** *export and import commands*

Element	Purpose	Key considerations
-E	Exports a single backup generation to the Informix Primary Storage Manager external pool	Use this option only if you set up a Informix Primary Storage Manager external pool. If you export a backup generation, you must specify a prefix, which identifies the exported backup. The **onsmsync** utility creates a subdirectory that includes the prefix in the external pool, and deposits the exported objects in that directory.
-g *generation*	Specifies the backup generation to export.	The default value is the current backup.
-I	Imports a single backup generation from the external Informix Primary Storage Manager pool.	If you import a backup generation from an external pool, you must specify the prefix for exported backup. The prefix identifies the backup generation that you want to import.
-l *log_ID*	Exports the backup generation that includes a logical log ID.	
-p *prefix*	Specifies the prefix that you are assigning to a backup generation that you are exporting or are using to identify the backup that you want to import.	When the **onsmsync** utility exports a backup generation, it uses the prefix as the name of the subdirectory in which it places the backup.
-t *timestamp*	Specifies a backup generation that includes a particular date and time (used only for exporting).	Use the ANSI or GLS_DATETIME format. You might want to include the date and time when you roll out a new version of an application.

Usage

If no options are specified, the **onsmsync** command synchronizes the **sysutils** database and emergency boot file with the storage-manager catalog. The **onsmsync** utility compares the backups in the **sysutils** database and emergency boot file with the backups in the storage-manager catalog and then removes all backups that are not in the storage manager catalog from the **sysutils** database and emergency boot file.

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 configuration parameter" on page 17-7.

The order of the commands does not matter except:

- The storage-space names or file name must come last.
- When exporting or importing, the **-E** or **-I** option must be first. For example, specify **onsmsync -E -g 2**, not **onsmsync -g 2 -E**.

Prerequisites for importing and exporting backup generations on different computers:

- You must have the same version of IBM Informix on the source and target computers and the computers must use the same operating system.
- You must set up the Informix Primary Storage Manager and create an external pool on the source and target computers.

When you use the **-E** or **-I** options to export or import a backup generation, you must specify the prefix that identifies the subdirectory where the backup generation is placed.

If you use the **-E** or **-I** options to export or import a backup generation, you cannot use any **onsmsync** command options that are not related to the export or import operation. For example, you cannot export a backup generation and regenerate the emergency boot file at the same time.

The **onsmsync -I** command renames your current `ixbar` file and creates a new file hat contains only the information that is necessary to restore the imported backup

You can use the **-cf** option with the **-g**, **-i**, or **-t** option to delete critical file backups from the storage manager.

If you apply the **-g** option and the **onsmsync** utility list of objects contains only logical logs and no space backups, the log backups are not expired. In this situation, use the **-t** or **-i** option to expire logical log backups.

Examples

The following example expires backups that started before November 30, 2012:

```
onsmsync -t "2012-11-30 00:00:00""
```

The following command exports the last backup generation to the directory call `gen` in the external pool:

```
onsmsync -E -p gen -g 1
```

The following command exports the fourth most recent backup generation to the directory called `gen` in the external pool:

```
onsmsync -E -p gen -g 4
```

The following command exports the current backup generation to the directory called `gen` in the external pool:

```
onsmsync -E -p gen
```

The following command exports all backup objects in generation 2 to the directory called `gen` in the external pool:

```
onsmsync -E -p gen -g 2
```

The following command exports all backup objects that have a timestamp of `2012-12-31 12:00:00` to the directory called `gen` in the external pool:

```
onsmsync -E -p gen -t "2012-12-31 12:00:00"
```

The following command imports all objects in the subdirectory that is identified by the prefix `gen`:

```
onsmsync -I -p gen
```

The following command imports all backup objects that were exporting using the prefix `gen` and the timestamp of `2012-12-31 12:00:00`. Because the prefix identifies the backup generation, you do not specify a timestamp.

```
onsmsync -I -p gen
```

The following command deletes all but the last two generations of critical file backups:

```
onsmsync -g 2 -cf yes
```

Related concepts:

"Managing storage devices" on page 15-9

"Informix Primary Storage Manager file-naming conventions" on page 15-19

Related tasks:

"Examples: Manage storage devices with Informix Primary Storage Manager" on page 15-3

"Restoring when a backup is missing data" on page 5-16

"Performing a cold restore" on page 6-9

Related reference:

"ON-Bar script" on page 3-5

Regenerate the emergency boot file

To regenerate the emergency boot file only, use the **onsmsync -b** command.

The **onsmsync -b** command saves the old emergency boot file as `ixbar.`*`server_number.system_time`* and regenerates it as `ixbar.`*`server_number`*.

Regenerate 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**.

Restriction: 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.

Delete 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 "onbar -v syntax: Verifying backups" on page 5-12.

Expire 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 00:00:00"
```

Expire 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**

Expire 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"
```

Expire backups with multiple point-in-time restores

If you perform more than one point-in-time restores, multiple timelines for backups exist.

The following figure shows three timelines with their backups.

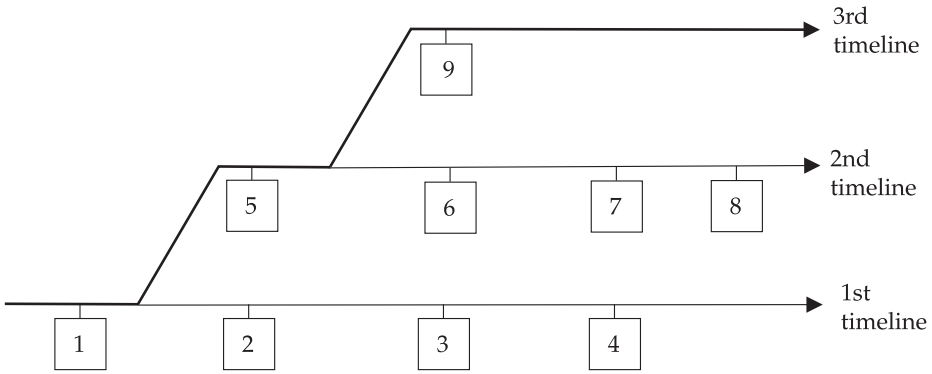


Figure 8-1. Multiple timelines for backups

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 the **onsmsync** utility 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 the previous figure, 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.

Expire all backups

The **onsmsync** utility retains the latest level-0 backup unless you use the `-0` option. If you use the `-0` and `-t` options, all backups from before the specified time are removed even if they are needed for restore. If you use the `-0` 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
```

Important: If you use the -O option with the -t, -i, or -g options, you might accidentally delete critical backups, making restores impossible.

Monitor 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 IBM Informix.
- Amount of data transferred to or from the XBSA API.
- Amount of data transferred to or from the Archive API.

Set 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 the onconfig file.

For example, the BAR_PERFORMANCE 1 setting displays the time spent transferring data between the IBM Informix instance and the storage manager.

See "BAR_PERFORMANCE configuration parameter" on page 17-10 for information about the options for this parameter.

View ON-Bar backup and restore performance statistics

To view ON-Bar performance results, open the ON-Bar activity log file, bar_act.log.

The location of the bar_act.log file is set by the BAR_ACT_LOG configuration parameter.

When the BAR_PERFORMANCE configuration parameter is set to 1 or 3, the activity report shows a transfer rate report.

```
2013-06-03 15:38:02 8597  8595 Begin restore logical log 310 (Storage Manager
                               copy ID: 28206 0).
2013-06-03 15:38:03 8597  8595 Completed restore logical log 310.
2013-06-03 15:38:08 8597  8595 Completed logical restore.
2013-06-03 15:38:19 8597  8595 PERFORMANCE INFORMATION

                                                     TRANSFER 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 |
-------------------------------------------------------------------------------------------------------------------------------
2013-06-03 15:38:19 8597  8595 PERFORMANCE INFORMATION

                                                     PERFORMANCE CLOCKS
-------------------------------------------------------------------------------------------------------------------------------
|                      ITEM DESCIRPTION                              |                    TIME SPENT                           |
-------------------------------------------------------------------------------------------------------------------------------
| Time to Analyze ixbar file                                        |                                                   0.000 |
------------------------------ ------------------------------------------------------------------------------------------------
```

Figure 8-2. Sample transfer rate performance in the ON-Bar activity log

When the BAR_PERFORMANCE configuration parameter is set to 2 or 3, the activity report has microsecond timestamps.

```
2013-06-03 16:34:04 15272  15270 /usr/informix/bin/onbar_d complete,
                                  returning 0 (0x00)
2013-06-03 16:45:11.608424 17085  17083 /usr/informix/bin/onbar_d -r -w
2013-06-03 16:46:07.926097 17085  17083 Successfully connected to Storage Manager.
2013-06-03 16:46:08.590675 17085  17083 Begin salvage for log 311.
2013-06-03 16:48:07.817487 17085  17083 Completed salvage of logical log 311.
2013-06-03 16:48:08.790782 17085  17083 Begin salvage for log 312.
2013-06-03 16:48:10.129534 17085  17083 Completed salvage of logical log 312.
2013-06-03 17:06:00.836390 17085  17083 Successfully connected to Storage Manager.
                                          ...
2013-06-03 17:07:26.357521 17085  17083 Completed cold level 0 restore datadbs07.
2013-06-03 17:07:28.268562 17085  17083 Begin cold level 0 restore datadbs08
  (Storage Manager copy ID: 28122 0).
2013-06-03 17:07:29.378405 17085  17083 Completed cold level 0 restore datadbs08.
```

Figure 8-3. Sample processing rates, in microseconds, in the ON-Bar activity log

Related concepts:

"bar_act.log file: ON-Bar activity log" on page 3-4

Related reference:

"BAR_ACT_LOG configuration parameter" on page 17-3

"BAR_PERFORMANCE configuration parameter" on page 17-10

Chapter 9. ON-Bar catalog tables

These topics describe 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.

The bar_action table

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.

Table 9-1. **bar_action** *table columns*

Column name	Type	Explanation
act_aid	SERIAL	Action identifier. A unique number within the table. Can be used with act_oid column 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.
act_status	INTEGER	Identifies the result of the action: `0` if successful, otherwise an ON-Bar-specific error code. For more information, see Chapter 10, "ON-Bar messages and return codes," on page 10-1.
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.

The bar_instance table

The **bar_instance** table contains descriptions of each object that is backed up.

ON-Bar writes a record to the **bar_instance** table for each successful backup. 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.

Table 9-2. **bar_instance** *table columns*

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_time	INTEGER	Timestamp (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. 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.
seal_time	INTEGER	The time that the log file was sealed after a backup completed.
prev_seal_time	INTEGER	The time that the previous log file was sealed.
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	INTEGER	Not used.
ins_first_log	INTEGER	In a standard backup, identifies the first logical log required to restore from this backup.
ins_chpt_log	INTEGER	The logical log that contains the archive checkpoint in the dbspace backup.
ins_last_log	INTEGER	In a standard backup, identifies the last logical log required to restore from this backup.

Table 9-2. **bar_instance** *table columns (continued)*

Column name	Type	Explanation
ins_partial	INTEGER	Partial flag from salvage.
ins_sm_id	INTEGER	Not used.
ins_sm_name	CHAR(128)	Not used.
ins_verify	INTEGER	Value is 1 if the backup is verified. Value is 0 if the backup is not verified.
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.
ins_backup_order	INTEGER	The order in which backups occur.

The bar_ixbar table

The **bar_ixbar** table, which stores a history of all unexpired successful backups in all timelines, is maintained and used by the **onsmsync** utility only.

The schema of the **bar_ixbar** table is identical to the schema of the **bar_syncdeltab** table, except for its primary key.

Table 9-3. **bar_ixbar** *table columns*

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.

Table 9-3. **bar_ixbar** *table columns (continued)*

Column name	Type	Explanation
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.
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_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 back up 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.
ixb_obj_type	CHAR(2)	Backup object type: **CD** critical dbspace **L** logical log **ND** noncritical dbspace or sbspace **R** rootdbs **B** blobspace

Related reference:

"The bar_syncdeltab table" on page 9-5

The bar_object table

The **bar_object** table contains descriptions of 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.

Table 9-4. **bar_object** *table columns*

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.
obj_type	CHAR(2)	Backup object type: **CD** critical dbspace **L** logical log **ND** noncritical dbspace or sbspace **R** rootdbs **B** blobspace

The bar_server table

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

Table 9-5. **bar_server** *table columns*

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.

The bar_syncdeltab table

The **bar_syncdeltab** table is maintained and used by the **onsmsync** utility only. This table is empty except when **onsmsync** is running.

The schema of the **bar_syncdeltab** table is identical to the schema of the **bar_ixbar** table, except for its primary key.

Related reference:

"The bar_ixbar table" on page 9-3

ON-Bar catalog map

This topic contains an example mapping between ON-Bar tables.

The following figure maps the ON-Bar tables on IBM Informix. In this figure, the gray lines show the relations between tables. The arrows show that the **ins_req_aid** value must be a valid **ins_aid** value.

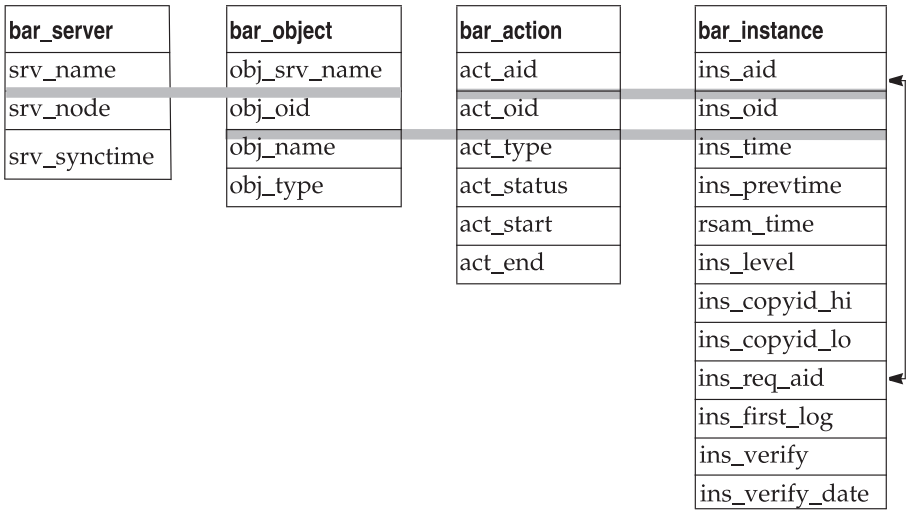


Figure 9-1. ON-Bar catalog map on Informix

Chapter 10. ON-Bar messages and return codes

ON-Bar prints informational, progress, warning, and error messages to the ON-Bar activity log file. ON-Bar return codes indicate the status of the command.

For a description of an error message, use the **finderr** utility or go to http://pic.dhe.ibm.com/infocenter/informix/v121/topic/com.ibm.em.doc/errors.html.

Related concepts:

"bar_act.log file: ON-Bar activity log" on page 3-4

Related reference:

"onbar -m syntax: Monitoring recent ON-Bar activity" on page 5-8

Message format in the ON-Bar message log

Messages in the ON-Bar activity log file contain timestamps, process IDs, and explanatory text.

A message in the ON-Bar activity log file, `bar_act.log`, has the following format:

timestamp process_id parent_process_id message

The following table describes each field in the message. No error message numbers appear in the ON-Bar activity log.

Table 10-1. 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.

Timestamps when storage managers hang

If a storage manager process hangs, the timestamp for the process in the ON-Bar activity log is inaccurate. An asterisk symbol is appended to the timestamp and the message indicates how long the storage manager process hung. The following example shows that the storage manager process hung for two minutes starting at 10:27. At 10:29, the storage manager completed the backup.

```
2013-02-26 10:27:10* 13410  25695 (-43085) WARNING: BAR_TIMEOUT Storage
                         Manager Progress may be stalled for at least 2 minutes.

2013-02-26 10:29:12 13410  25695 Completed level 0 backup dbspace1 (Storage Manager
                                 copy ID: 1509564809 0).
```

Related reference:

"onbar -m syntax: Monitoring recent ON-Bar activity" on page 5-8

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 in the subdirectory for your locale under the `$INFORMIXDIR/msg` directory.

ON-Bar message type	Message numbers
ON-Bar usage	-43000 to -43007 and -43357
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
Back up 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 return codes

You can troubleshoot problems by viewing activity log messages that are applicable for particular return codes.

The following table shows the ON-Bar return codes for all IBM Informix database servers. These return codes are accompanied by messages in the ON-Bar activity log. For details about an error, review the activity log before you call Technical Support.

Table 10-2. 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.

Table 10-2. Common ON-Bar return codes (continued)

Decimal value	ON-Bar return code description
104	Adstar Distributed Storage Manager (ADSM) is in generate-password mode. ON-Bar does not support ADSM running in generate-password mode. For information about 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.
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 try the restore again.
121	ON-Bar was unable to determine the list of dbspaces.
122	Deadlock detected. The ON-Bar command is contending with another process. Try the ON-Bar command again.
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.
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
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.

Table 10-2. Common ON-Bar return codes (continued)

Decimal value	ON-Bar return code description
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 try the backup or warm restore again.
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.
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. Try the fake backup with the **onbar -b -F** command again. Only IBM Informix 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 try the ON-Bar command again.

Table 10-2. Common ON-Bar return codes (continued)

Decimal value	ON-Bar return code description
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 stopped because one or more blobspaces were down. Attempt to restore the blobspace. If the restore fails, try the log backup with the **onbar -l -O** command again. 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 try the ON-Bar command again.
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.
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 try the command with quotes around the *datetime* value again. 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.

Table 10-2. Common ON-Bar return codes (continued)

Decimal value	ON-Bar return code description
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 `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.
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 try the **onbar-r -e** command again to complete the external restore. If that does not work, try the external restore from a different external backup.

Table 10-2. Common ON-Bar return codes (continued)

Decimal value	ON-Bar return code description
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.
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 `-0` 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 `-0` 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 `-0` 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. Try the restore without the `-0` option again.
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 `-0` option. If you used the `-0` 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.
247	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 try the cold restore again. If the cold restore still fails, contact Technical Support.
252	For assistance, contact Technical Support.

Part 3. ontape backup and restore system

Related reference:

"LTAPEBLK configuration parameter" on page 17-16
"LTAPEDEV configuration parameter" on page 17-17
"LTAPESIZE configuration parameter" on page 17-18
"TAPEBLK configuration parameter" on page 17-21
"TAPEDEV configuration parameter" on page 17-21
"TAPESIZE configuration parameter" on page 17-23

Chapter 11. Configure ontape

You configure the **ontape** utility by setting configuration parameters for backups of storage spaces and logical logs.

You can also set the IFX_BAR_USE_DEDUP environment variable to optimize the backup images for deduplication devices.

Related reference:

"IFX_BAR_USE_DEDUP environment variable" on page 17-15

"Comparison of the ON-Bar and ontape utilities" on page 1-8

Set configuration parameters for the ontape utility

The **ontape** utility uses eight configuration parameters in the `onconfig` file. Two of the configuration parameters specify filter programs for transforming data during backup and restore; the other six are used to create storage-space and logical-log backups.

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*.

Data transformation filter parameters for ontape

The BACKUP_FILTER and RESTORE_FILTER configuration parameters specify the names of external programs that you can use to transform data before backup and after a restore.

BACKUP_FILTER
Specifies the location and name of an external filter program used in data transformation. This filter transforms data before 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 before the backup, such as extracting 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.

See "BACKUP_FILTER configuration parameter" on page 17-2 and "RESTORE_FILTER configuration parameter" on page 17-20 for syntax and usage information, which is the same for ON-Bar and **ontape**.

Tape and tape device parameters for ontape

The first set of configuration 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** writes storage space data during an archive and the source from which **ontape** reads data during a restore.

To configure **ontape** to use **stdio**, set TAPEDEV to STDIO.

When backing up to or restoring from a cloud environment, use the following syntax for the TAPEDEV configuration parameter:

```
TAPEDEV 'local_path, keep=option, cloud=cloud_vendor, url=url'
```

- **local_path** is the complete path name of the directory where storage spaces backup objects are stored temporarily.
- **option** can be set to *yes* or *no*. If keep is set to *yes*, the ontape utility retains the backup objects in the local directory. If **keep** is set to *no*, the backup objects are deleted after they are transferred to or from the cloud storage location.
- **cloud_vendor** is the name of the cloud storage vendor.
- **url** is the cloud storage location where the storage space backup data is stored persistently.

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 - 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.

When backing up to or restoring from a cloud environment, use the following syntax for the LTAPEDEV configuration parameter:

```
LTAPEDEV 'local_path, keep=option, cloud=cloud_vendor, url=url'
```

- **local_path** is the complete path name of the directory where log backup objects are stored temporarily.
- **option** can be set to *yes* or *no*. If keep is set to *yes*, the ontape utility retains the backup objects in the local directory. If **keep** is set to *no*, the backup objects are deleted after they are transferred to or from the cloud storage location.
- **cloud_vendor** is the name of the cloud storage vendor.
- **url** is the cloud storage location where the log backup data is stored persistently.

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 - 2,097,151.

The following topics contain information about how to set the tape-device, tape-block-size, and tape-size parameters for both storage-space and logical-log backups.

Related tasks:

"Back up to Amazon Simple Storage Service" on page 12-10

Set 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.
- Configure parameters to perform backup to a cloud.

The following sections explain each of these points.

Related tasks:

"Back up to Amazon Simple Storage Service" on page 12-10

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.

- Stop 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.

Specify 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 back up to a disk file while using the tape device for a backup.

Specify 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 LTAPEDEV and TAPEDEV 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 "Rename existing files" on page 12-6.

Specify 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 on UNIX and Linux platforms.

You should not do a continuous backup to a remote device.

The remote device and the database server computer must have a trusted relationship so that the **rsh** or the **rlogin** utility can connect from the database server computer to the remote device computer without asking for password. You can establish a trusted relationship by configuring the `/etc/hosts.equiv` file, the `~/.rhosts` file, or any equivalent mechanism for your system on the remote device computer. If you want to use a different utility to handle the remote session than the default utility used by your platform, you can set the **DBREMOTECMD** environment variable to the specific utility that you want to use.

To specify a tape device on another host computer, use the following syntax to set the TAPEDEV or LTAPEDEV configuration parameter:

host_machine_name:tape_device_pathname

The following example specifies a tape device on the host computer `kyoto`:

```
kyoto:/dev/rmt01
```

For information about the tape size for remote devices, see “Tape size for remote devices” on page 11-6.

Related reference:

DBREMOTECMD environment variable (UNIX) (SQL Reference)

Specify /dev/null for a tape device

A best practice is to not use `/dev/null` as the device when backing up. However, if you decide that you do not need to recover transactions from the logical log, you can specify `/dev/null` as a tape device for logical-log backups

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`.

When you specify the tape device as `/dev/null`, block size and tape size are ignored. If you set the LTAPEDEV configuration parameter either to or from `/dev/null`, you must restart the database server for the new setting to take effect.

Important: When you set the LTAPEDEV configuration parameter 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.

Set TAPEDEV to stdio

To configure the **ontape** utility to read from standard input or write to standard output, set the TAPEDEV configuration parameter to **stdio**.

Rewind 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.

Specify the tape-block-size

Use the TAPEBLK and LTAPEBLK configuration parameters to set 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.

Specify the tape size

Use the TAPESIZE and LTAPESIZE configuration parameter parameters to 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 - 2,097,151.

Tape size for remote devices

You can estimate the amount of continuous logical-log backup data that can be stored on tape on 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.

Restriction: You cannot set tape size to 0 for remote devices.

Changing your ontape configuration

You can change the values of the TAPEDEV, TAPEBLK, and TAPESIZE or LTAPEDEV, LTAPEBLK, and LTAPESIZE configuration parameters after performing a backup and reviewing the current and new parameter values.

Prerequisites: Before you change the parameters for **ontape**:

- Perform a level-0 backup.
- Examine your configuration file (the file specified in `$INFORMIXDIR/etc/$ONCONFIG`) by executing **onstat -c** while the database server is running.
- If you plan to change TAPEDEV or LTAPEDEV to a different tape device, verify that the tape device can read the block size that you specify with the TAPEBLK or LTAPEBLK configuration parameter. If not, you cannot restore the tape. (The **ontape** utility does not check the tape device when you specify the block size.)
- Be sure that you are logged in as user **root** or **informix**.

You can change the values of parameters for **ontape** while the database server is online.

To change the value of TAPEDEV, TAPEBLK, and TAPESIZE or LTAPEDEV, LTAPEBLK, and LTAPESIZE:

1. From the command line, use a text editor to edit your `onconfig` file.
2. Save the file.

The change takes effect immediately. However, if you set either the TAPEDEV parameter or the LTAPEDEV parameter to `/dev/null`, you must restart the database server.

Related reference:

"LTAPEBLK configuration parameter" on page 17-16
"LTAPEDEV configuration parameter" on page 17-17
"LTAPESIZE configuration parameter" on page 17-18
"TAPEBLK configuration parameter" on page 17-21
"TAPEDEV configuration parameter" on page 17-21
"TAPESIZE configuration parameter" on page 17-23

Chapter 12. Back up with ontape

These topics describe how to use the **ontape** utility 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:

- "Change database logging status"
- "Create a backup" on page 12-2
- "Starting a continuous logical-log file backup" on page 12-14
- "ontape utility syntax: Perform a restore" on page 13-3
- "Use external restore commands" on page 14-4

Start ontape

When you need more than one tape during a backup, the **ontape** utility prompts for each additional tape.

If the database server is in maintenance mode, for example, during a conversion, then the **ontape** utility 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**)

Restriction: Do not start the **ontape** utility in background mode (that is, with 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 attempts to retry. When **ontape** expects a yes-or-no response, it assumes that any response not recognized as a "yes" is "no".

Exit codes for ontape

The **ontape** utility has the following two exit codes:

0 Indicates a normal exit from **ontape**.

1 Indicates an exception condition.

Change database logging status

You can use the **ontape** utility to change the logging status of a database. Most changes in logging mode require a full level-0 backup.

You cannot change the logging mode of an ANSI-compliant database.

You can change an unbuffered logged or buffered logged database to an unlogged database without making a backup.

You can make the following logging modes changes with a level-0 backup:

- An unbuffered logged or buffered logged database to an ANSI database
- An unbuffered logged database to a buffered logged database
- A buffered logged database to an unbuffered logged database

Examples

The following command changes the logging mode of a database named **stores7** to unbuffered logging:

```
ontape -s -L 0 -U stores7
```

The following command changes the logging mode of a database to ANSI-compliant logging:

```
ontape -s -L 0 -A stores7
```

The following command changes the logging mode of a database to unlogged:

```
ontape -N stores7
```

Related reference:

"ontape utility syntax: Perform a backup" on page 12-7

Database-logging status (Administrator's Guide)

Create a backup

These topics explain how to plan for and create backups of your database server data.

Backup levels that ontape supports

The **ontape** utility supports level-0, level-1, and level-2 backups.

For information about scheduling backups, see "Plan 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 backup

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 backup

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 backup

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 change the physical schema.

Perform a level-0 backup after you make the following administrative changes:

- Changing the TAPEDEV or LTAPEDEV configuration parameter 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

Consider waiting to make these changes until your next regularly scheduled level-0 backup.

Tip: Although you no longer need to back up 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 temp tables during heavy activity.
- Make sure enough logical-log space exists.
- Keep a copy of your configuration file.
- Verify consistency before a level-0 backup.
- 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.

Avoid 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 stops and your monitor displays a NO FREE DISK error message.

Make 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 configuration 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 do 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 about 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.

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 stop the backup. Thus, make sure that an operator is available.

After a tape fills, the **ontape** utility rewinds the tape, displays the tape number for labeling, and prompts the operator to mount the next tape when you need another one. Follow the prompts for labeling and mounting new tapes. A message informs you when the backup is complete.

Label tapes created with ontape:
When you label tapes created with the **ontape** utility, 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

A backup to standard output creates an archive in the memory buffer provided by the operating system. If you choose to back up to standard output, you do not need to provide tapes or other storage media.

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 directory of the output 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 by using standard I/O” on page 13-14.

Related reference:

“Restore from standard input” on page 13-13

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.

Set 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.

Rename 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 `servername_YYYYMMDD_hhmmss_archive-level`.
- Logical log backup files

  The backup time is added, and has the format `servername_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.

Override 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 file name, *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`.

ontape utility syntax: Perform a backup

Use **ontape** utility command options to back up to tape.

Pre-requisites: Before you begin a backup, perform the following steps:

- If necessary, place a write-enabled tape on the tape-drive device that TAPEDEV specifies.

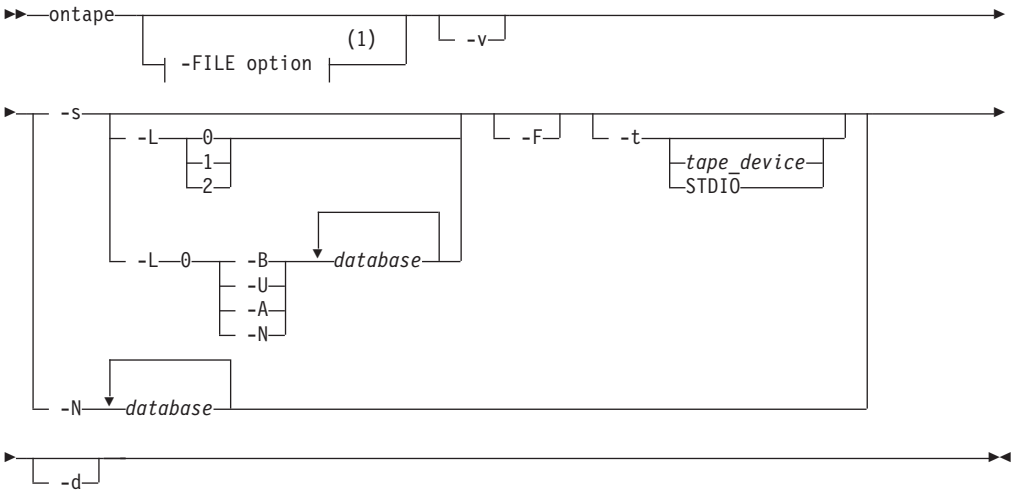
  If you set TAPEDEV to STDIO, ensure that there is enough memory for the backup data.
- If you are using a tape device, put the device online with the appropriate operating-system command.
- Place the database server in online or quiescent mode.
- Log in as the owner of the database server: user **informix** or **root** for a standard installation, or the owner of the non-root installation.

If you are using TAPE devices, do not store more than one backup on the same tape. The tape devices must be of the rewindable type. 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.

Syntax

Create a backup



Notes:

1 See The -FILE option.

Element	Purpose	Key considerations
-A	Directs **ontape** to change the status of the specified database to ANSI-compliant logging.	A database that has ANSI-compliant logging cannot be changed to a different logging mode. A level-0 backup is required to make this logging mode change.
-B	Directs **ontape** to change the status of the specified database to buffered logging.	A level-0 backup is required to make this logging mode change.
-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.
database	The name of the database to change the logging mode.	The database name cannot include a database server name. More than one database name can be specified in the same command.
-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 a high-availability cluster. 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 Administrator's Reference* 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.
-N	Directs **ontape** to end logging for the specified database.	A backup is optional with this logging mode change.
-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	Directs **ontape** to use a different tape device for the current backup or restore.	The `-t` option overrides the value of the TAPEDEV configuration parameter for the current backup or restore. The `-t STDIO` option directs **ontape** to back up to standard output or restore from standard input.
tape_device	The name of the tape device on which to store the backup.	
-U	Directs **ontape** to change the status of the specified database to unbuffered logging.	A level-0 backup is required to make this logging mode change.
-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.

The **ontape** utility backs up the storage spaces in the following order: root dbspaces, blobspaces, sbspaces, and dbspaces.

Related reference:
"Change database logging status" on page 12-1

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. The following figure 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 12-1. 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.

Back 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 then alter raw tables to standard tables before performing transactions. For more information, see the *IBM Informix Administrator's Guide*.

Back up to Amazon Simple Storage Service

You can use the **ontape** utility to back up and restore data to or from the Amazon Simple Storage Service (S3). You are responsible for terms and any charges associated with your use of the Amazon Simple Storage Service.

Prerequisites:

- You must have an Amazon account to perform cloud storage backups. See the Amazon website for instructions about setting up an account.
- Java™ version 1.5 or later is required.
- Backup objects must be 5 GB or smaller.

The following steps show how to back up data to the Amazon Simple Storage Service (S3) System and restore from it by using **ontape** backup and restore utility. In this context, cloud storage refers to an online storage service over the Internet. If you choose to back up to cloud storage, you do not need to provide tapes. Instead, you back up the data to a virtual device, most likely located on the Internet.

1. Configure the online storage device.
   a. Using a web browser, navigate to the Amazon S3 website and log on.
   b. Obtain an access key ID and a secret access key.
   c. Store the access credentials in a file. Set the permissions on the file to allow access only to user executing the **ontape** utility.
      - On UNIX systems, store the values in the file: `$INFORMIXDIR/etc/ifxbkpcloud.credentials`
      - On Windows systems, store the values in: `%INFORMIXDIR%\etc\ifxbkpcloud.credentials`

      The file must have the following format:

      ```
      secretKey=secret_access_key
      accessKey=access_key_ID
      ```
   d. Use the **ifxbkpcloud.jar** utility to create and name a storage device in the region where you intend to store Informix data. Amazon uses the term *bucket* to describe the container for backup data. The storage device name you choose has the same restrictions as those for the bucket name in Amazon S3 and must be unique.

      For example, the following command creates a storage device named **mytapedevice** at a **US Standard** region on Amazon S3. Run the command from the **$INFORMIXDIR/bin** directory on UNIX systems, or from **%INFORMIXDIR%\bin** on Windows systems.

      ```
      java -jar ifxbkpcloud.jar CREATE_DEVICE amazon mytapedevice US_Standard
      ```
2. Set the TAPEDEV and LTAPEDEV configuration parameters in the `onconfig` file to point to the cloud storage location. For example:

   ```
   TAPEDEV '/opt/IBM/informix/tapedev_dir, keep = yes, cloud = amazon,
           url = https://mytapedevice.s3.amazonaws.com'
   LTAPEDEV '/opt/IBM/informix/ltapedev_dir, keep = yes, cloud = amazon,
           url = https://mylogdevice.s3.amazonaws.com'
   ```
3. Back up data to the online storage device by using the **ontape** utility.

   ```
   ontape -s -L 0
   ```

   You can restore data from the cloud storage by using the following command:

   ```
   ontape -r
   ```

   You should use https secure data transmission when transferring data to cloud storage. You should encrypt data before transferring data to a cloud image. To encrypt data, use the BACKUP_FILTER and RESTORE_FILTER configuration

parameters to call an external encryption program. The archecker utility does not support table-level restore of data from cloud storage.

Related reference:

"Tape and tape device parameters for ontape" on page 11-2

"Set the tape-device parameters" on page 11-3

"Cloud storage file naming conventions"

"The **ifxbkpcloud.jar** utility"

The ifxbkpcloud.jar utility

Use the **ifxbkpcloud.jar** utility to configure an online storage device for the Amazon Simple Storage Service.

The following options are supported by the **ifxbkpcloud.jar** utility:

- CREATE_DEVICE *provider device* [*region*]
- DELETE_DEVICE *provider device*
- LIST_DEVICES *provider*
- DELETE_FILE *provider device file*
- LIST_FILES *provider device*

The parameters for the **ifxbkpcloud.jar** commands are defined as follows:

- *provider* is **amazon**.
- *device* is the name of the storage device.
- *region* is one of the following: **US_Standard**, **US_West**, **EU_Ireland** or **AP_Singapore**.
- *file* is the name of backup object (key) stored on Amazon S3.

Error messages from **ifxbkpcloud.jar** are written to $INFORMIXDIR/ifxbkpcloud.log on UNIX machines and to %INFORMIXDIR%\ifxbkpcloud.log on Windows machines.

Related tasks:

"Back up to Amazon Simple Storage Service" on page 12-10

Cloud storage file naming conventions

Files associated with cloud storage backups have unique file names.

Data space backup files are saved by using the following format:

hostname_servernum_Larchive_level

Log backup file names are saved by using the following format:

hostname_servernum_lognnnnnnnnnn

If the object exists at the cloud storage location, the file is renamed to avoid overwriting old object. Renaming the file adds a timestamp to the object name.

Data space backup files are saved by using the following format:

hostname_servernum_YYYYMMDD_hhmmss_Larchive_level

Log backup file names are saved by using the following format:

*hostname_servernum_lognnnnnnnnnn*_YYYYMMDD_hhmmss

Related tasks:

"Back up to Amazon Simple Storage Service" on page 12-10

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 the owner of the database server: user **informix** or **root** for a standard installation, or the owner of the non-root installation.

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 "Create a backup" on page 12-2.

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 stop the backup (by 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 "Start an automatic logical-log backup" on page 12-14 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 "Monitor backup history by using oncheck," you know the backup completed.

Monitor backup history by using oncheck

You can monitor the history of your last full-system backup by using **oncheck**.

Execute the **oncheck -pr** command 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.

Back 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 about these issues, see "Logical-log backup" on page 1-2.

Use blobspace TEXT and BYTE data types and logical-log files

You must keep in mind the following two points 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 about switching logical-log files, see your *IBM Informix Administrator's Guide*.

Use /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`.

Important: 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 restore" on page 13-1.) 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 about monitoring the status of logical-log files, see your *IBM Informix Administrator's Guide*.

Start 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.

Request 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 **ontape -a** command.

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.

To start a continuous backup of the logical-log files, use the **ontape -c** command. 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.

Whether the logical-log files are backed up to tapes or a directory depends on the setting of the LTAPEDEV configuration parameter:

- If the LTAPEDEV configuration parameter is set to a tape device, someone must always make media available for the backup process. When the specified mounted tape becomes full before the end of the logical-log file, the database server prompts the operator for a new tape. Also, you must dedicate the backup device to the backup process.
- If the LTAPEDEV configuration parameter is set to a directory, logical-log files can be backed up unattended. Logical logs are backed up as they fill and a new file is created in the directory for each logical log. Backup is limited by space available for new files.

To back up to a directory, as an alternative to using the **ontape -c** command, you can call the **ontape -a -d** automatic logical log backup command from a script specified by the ALARMPROGRAM configuration parameter. You can use either the `alarmprogram` or script or the `log_full` script, both of which are found in the `$INFORMIXDIR/etc` directory.

To use the `alarmprogram` script to back up logical logs to a directory:

1. Set the LTAPEDEV parameter to an existing directory. Make sure that 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.

End 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 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 (by 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 configuration 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 "Use /dev/null when you do not need to recover" on page 12-13.

  If the log backup device on any server node in a high-availability cluster is set to `/dev/null` (on Linux or UNIX) or NUL (on Windows), then the backup device for all of the other servers within the cluster (including the primary server and any HDR, RSS or SDS secondary servers) must also be set to `/dev/null` (or NUL).
- 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 13. Restore with ontape

These topics provide instructions for restoring data with the **ontape** utility 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.

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 IBM Informix. 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 1-5.

Full-system restore

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."

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.

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 "Rename chunks during a restore" on page 13-10. You can also rename chunks for an external cold restore; see "Rename chunks" on page 14-5 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.

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 about how the database server looks for temporary space, see the discussion of DBSPACETEMP in the *IBM Informix Administrator's Guide*.

Important: Make sure that 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.

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.

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.

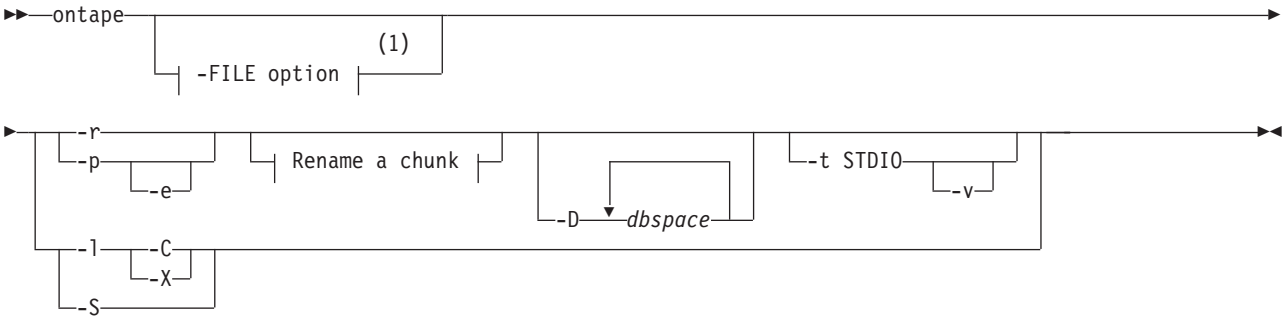
The dbspaces not restored during the cold restore do not become available until after the database server restores them during a warm restore, even though a critical dbspace possibly did not damage them.

ontape utility syntax: Perform a restore

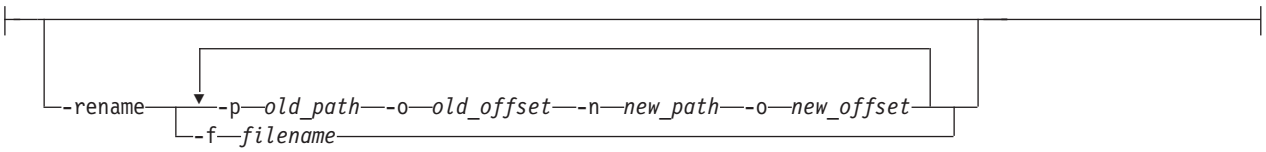
Use the **-r** option to perform a full physical and logical restore of the database server data with **ontape**. Use the **-D** option to restore selected storage spaces. Use the **-rename** option to rename chunks during the restore.

You must run the **ontape** command as the owner of the database server: user **informix** or **root** for a standard installation, or the owner of the non-root installation. The owner of the database server for the restore must be the same as the owner of the database server for the backup.

Syntax



Rename a chunk:



Notes:

1 See The -FILE option.

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.

Element	Purpose	Key considerations
-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 "Restore selected storage spaces" on page 13-8.
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 14, "Perform an external backup and restore," on page 14-1. This option is compatible with renaming chunks for external cold restores.
-f *filename*	Specifies a file containing the names and offsets of chunks to be renamed and their new locations. Use to rename many chunks at one time.	The file name can be any valid UNIX or Windows file name, 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`) file names. 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* The current path and file name of the chunk. *old_offset* The current offset of the chunk, in kilobytes. *new_path* The new path and file name of the chunk. *new_offset* 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 about renaming chunks during a restore, see "Rename chunks during a restore" on page 13-10.
-S	Directs **ontape** to perform a logical log salvage.	If you want to salvage logical logs, you must use the **-S** option before performing a restore from standard input. The LTAPEDEV configuration parameter must be set to the logical log tape device.

Element	Purpose	Key considerations
-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.

Restore the whole system

This section outlines the prerequisites and steps you need to complete to restore your entire database server with **ontape**.

The following list summarizes the main steps in a full-system restore:

1. Gather the appropriate backup and logical log 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.

Gather backup and logical-log tapes before restoring

Before you restore an entire database system, you must gather backup and logical log tapes. If you changed the names of backup and logical log files, you must also manually rename the files to their original file names.

Backup tapes

Gather 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

If at the time of the archive checkpoint, an open transaction started, gather all logical-log tapes before you perform the level-0 backup.

Gather all the logical-log tapes from the backup after the latest level-0 backup of the storage spaces you are restoring.

When using **ontape** to create an archive backup of the system, a snapshot of the logical logs is included with the archive. At the end of the archive, the system displays a message that indicates what logical logs are included in the archive. The snapshot is included in the archive so that if any open transactions exist at the time of the backup, those transactions can be reconciled when the archive is restored. Then:

- If you decide not to replay any logical logs, the system can be brought to a consistent state.

- If you decide to replay logical logs, the logs contained within the archive backup are discarded and you must replay transactions from the logical log backups.

  The starting log file is the oldest log file containing an open transaction at the time of the last restored archive. You can identify that log file from the message that was displayed when the last archive was restored.

Example:

- The **ontape -s -L 0** command performs a level-0 backup of the system and displays a message that states that the archive contains logs 2-4.
- The **ontape -s -L 1** command performs a level 1 incremental backup of the system and displays a message that states that the archive contains logs 8-9.

If you restore only the level-0 archive and want to replay logs, you need log backups starting with log 2. If you restore the level-0 and level-1 archives and want to replay logs, you need log backups starting with log 8.

The restore of the logical log files uses an archive format, not a log file format. However, the logs contained within the restored archive are in log file format, not an archive format.

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 configuration parameters. If files were renamed, including by **ontape** because of repeated archives and backups, you must manually rename the files 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 12-6 for the naming conventions of these files.

Decide on a complete cold or a mixed restore

As mentioned in "Cold, warm, or mixed restores" on page 13-1, 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 cold restore or a mixed restore.

Verify 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

Set 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.)

Set 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 configuration file, MIRRORPATH and MIRROROFFSET.

Verify that the raw devices or files 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.

Perform a cold restore

To perform a cold restore with **ontape**, the database server must be offline.

You must run the **ontape** command as the owner of the database server: user **informix** or **root** for a standard installation, or the owner of the non-root installation. The owner of the database server for the restore must be the same as the owner of the database server for the backup.

Run 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
```

Salvage 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
...
```

Mount 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. The **ontape** utility 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 IBM Informix instance into quiescent mode.

Restore 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 "Back up logical-log files with ontape" on page 12-13.

To restore the logical logs, use the **ontape -l** command.

Bring 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 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.

Restore selected storage spaces

These topics outline 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.

Before you attempt a restore, familiarize yourself with these instructions.

The following list describes the main steps in a warm restore:

Gather 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.

Ensure 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.

Back 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 "Back up logical-log files with ontape" on page 12-13.

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.
```

Perform a warm restore

To perform a warm restore with **ontape**, the database server must operate in online or quiescent mode.

You must run the **ontape** command as the owner of the database server: user **informix** or **root** for a standard installation, or the owner of the non-root installation. The owner of the database server for the restore must be the same as the owner of the database server for the backup.

To restore selected storage spaces, run 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 "Restore the whole system" on page 13-5.

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.

Restore 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 "Back up raw tables" on page 12-9 and the *IBM Informix Administrator's Guide*.

Configuring continuous log restore with ontape

Ensure that the version of IBM Informix is identical on both the primary and secondary systems.

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 1-8.

To configure continuous log restore with **ontape**:

1. On the primary system, perform a level-0 archive with the **ontape -s -L 0** command.
2. On the secondary system, copy the files or mount the tape (as assigned by LTAPEDEV) and perform a physical restore with the **ontape -p** command.
3. 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.
4. On the primary system, back up logical logs with the **ontape -a** command.
5. 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 **ontape -l -C** command.
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, use the **ontape -l** command.
   - After all available logical logs are restored, use the **ontape -l -X** command.

Related concepts:

"Continuous log restore" on page 1-8

Rename 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

restores 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 "ontape utility syntax: Perform a restore" on page 13-3.

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 "Rename chunks" on page 14-5 for more information.

Validation sequence for renaming chunks

During a cold restore, **ontape** performs the following validations to rename chunks:

- It validates that the old chunk path names and offsets exist in the archive reserved pages.
- It validates that the new chunk path names and offsets do not overlap each other or existing chunks.
- 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`.
- It restores the data from the old chunks to the new chunks (if the new chunks exist).
- 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.

Important:

- Perform a level-0 archive after you rename chunks; otherwise your next restore fails.
- 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.
- 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*.

Rename chunks with command-line options

To rename the chunks by supplying information about 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.

Rename chunks with a file

To rename the chunks by supplying a file named `listfile`, use the following 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.

Rename chunks while specifying other options

To rename the chunks with 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 by 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.

Rename 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

Renaming a chunk to a nonexistent device

To rename a chunk to a nonexistent device:

1. Rename the chunk: using the following command: **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** with the **ontape -r -D sbspace1** command.
6. Perform a level-0 archive.

Restore from standard input

You can perform a restore from standard input, you must first have performed a backup to standard output.

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 before 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.

Examples

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.

Related reference:

"Back up to standard output" on page 12-5

Restore 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 Informix 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 by 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.

On HDR, the secondary server can restore only level-0 archives.

To use standard I/O to perform the backup and restore, set the **TAPEDEV** configuration parameter to **STDIO**, or you can specify **-t STDIO** from the command line.

For example, if the **TAPEDEV** configuration parameter is set to **STDIO**, the following command loads data into the secondary server on an HDR pair (named **secondary_host**).

```
ontape -s -L 0 -F | rsh secondary_host "ontape -p"
```

In the next example, assume that the **TAPEDEV** configuration parameter is not set. The following command loads data into the secondary server of an HDR pair (named **secondary_host**):

```
ontape -s -L 0 -F -t STDIO | rsh secondary_host "ontape -t STDIO -p"
```

The examples perform a fake level-0 archive of the database server on the local computer, pipe the data to the remote computer by using the **rsh** system utility, and perform a physical restore on the remote computer by reading the data directly from the pipe.

Important: The previous examples require that the **INFORMIXDIR**, **INFORMIXSERVER**, **INFORMIXSQLHOSTS**, and **ONCONFIG** environment variables be 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 14. Perform an external backup and restore

These topics discuss performing an external backup and recovering data by restoring it with the **ontape** utility.

Recover data by using an external backup and restore

You can perform an *external backup and restore*, which eliminates the downtime of systems because the backup and restore operations are performed external to the IBM 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 vendor 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 14-3.

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 by 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.

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 a cold external restore" on page 14-5.

Rules for an external backup

Before you begin an external backup, keep in mind the following rules:

- 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 "Track an external backup" on page 7-3.

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 with the **onmode -c block** command. The system takes a checkpoint and suspends all update transactions. Users can access the database server in read-only mode.
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 with the **onmode -c unblock** command.
4. Back up all the logical logs including the current log so that checkpoint information is available for the external restore.

   Important: 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 **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. using the **ontape -a** command.

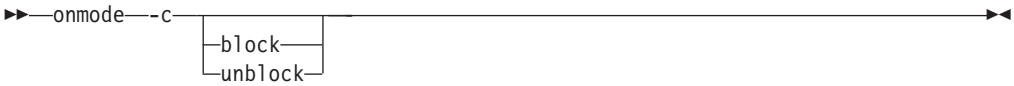
If you lose a disk or the whole system, you are now ready to perform an external restore.

Prepare for an external backup

These topics describe the commands used to prepare for an external backup. For the procedure, see "Performing an external backup."

Block and unblock Informix

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**

Track 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. The following table shows the items we recommend that you track in an external backup.

Table 14-1. 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	UNIX: `/work/dbspaces/rootdbs` Windows: `c:\work\dbspaces\rootdbs`
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 12.10

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.

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**).

Use 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.

Perform 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 "ontape utility syntax: Perform a restore" on page 13-3.

Rules for an external restore

Before you begin an external restore, know what you can and cannot restore from an external backup and be aware of the rules for an external restore.

These requirements and rules are:

- 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 with **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.

Rename chunks

You can rename chunks in an external cold restore by 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 a cold external restore

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 storage spaces.

To perform a cold external restore:

1. Shut down the database server with 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.
3. To perform an external restore of all storage spaces followed by a logical restore, use the following commands:
   - **ontape -p -e**
   - **ontape -l**

Examples of external restore commands

The following table contains an example of external restore commands.

External restore command	Action	Comments
ontape -p -e **ontape -l**	Physical external restore and logical restore	The system restores the logical logs from the oldest external backup.
ontape -p -e -rename -f	External cold restore with renamed chunks	

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 **onmode -c block** command.
2. Externally back up all chunks on the source database server.
3. When the backup completes, unblock the source database server with the **onmode -c unblock** command.
4. Make the source database server the primary server with the following command: **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 with the **ontape -p -e** command.

7. Make the target database server the secondary server with the following command: **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 with the **ontape -l** command.

   The `database server operational` messages appear in the message log on the primary and secondary servers.

Part 4. Informix Primary Storage Manager

The IBM Informix Primary Storage Manager manages storage for ON-Bar backup and restore operations, including parallel backups, that use file devices (disks).

Chapter 15. Informix Primary Storage Manager

IBM Informix Primary Storage Manager is an application that manages storage devices used for backup and restore requests that are issued by ON-Bar. This storage manager supports both serial and parallel processing for backup and restore requests.

Informix Primary Storage Manager consists of the following components:

onpsm utility
A command-line utility that you can use to perform the following tasks:

- Create, modify, and delete storage devices
- Define and modify the maximum sizes for devices
- Move backup information from one device to another within a device pool
- Determine whether volumes, storage objects, and devices are locked or busy
- Release locked volumes, storage objects, and devices
- Verify volume names and labels

XBSA shared library
A unique version of the X/Open Backup Services API (XBSA) shared library that ON-Bar and the Informix Primary Storage Manager use to communicate with each other. When ON-Bar stores or retrieves data that is stored on storage devices, the storage manager coordinates the request through the XBSA interface at the device level. You specify the location of the XBSA shared library with the BAR_BSALIB_PATH configuration parameter.

Storage catalog tables
A set of flat files that track information about all storage objects, devices, and device pools. These files are required to restore backup objects that are created by Informix Primary Storage Manager. By default, these files are stored in the `$INFORMIXDIR/etc/psm` directory. You can use the PSM_CATALOG_PATH configuration parameter to specify another location for the storage catalog tables.

Important:

- Back up the storage catalog tables with your operating system tools as part of a disaster recovery strategy. The storage catalog tables are not backed up with the database instance and they are not associated with Informix system catalog tables.
- To prevent the storage catalog tables from getting too large, delete old generations of backups regularly. Use the **onsmsync** utility to manage expiration policies.

The configuration parameters that you use to configure the Informix Primary Storage Manager are in the `onconfig` file.

You define and maintain storage devices with the **onpsm** command-line utility. You can configure one device at a time or generate a device-configuration file to configure multiple devices. During backups, Informix Primary Storage Manager

selects a device from a pool of available devices. If the device becomes full or fails, the storage manager automatically moves to another device in the same pool.

Informix Primary Storage Manager writes informational, warning and error messages to the storage manager activity log. You can use the PSM_ACT_LOG configuration parameter to specify the location of the activity log. If the PSM_ACT_LOG configuration parameter does not contain information, the storage manager puts activity information in the directory specified with the BAR_ACT_LOG configuration parameter.

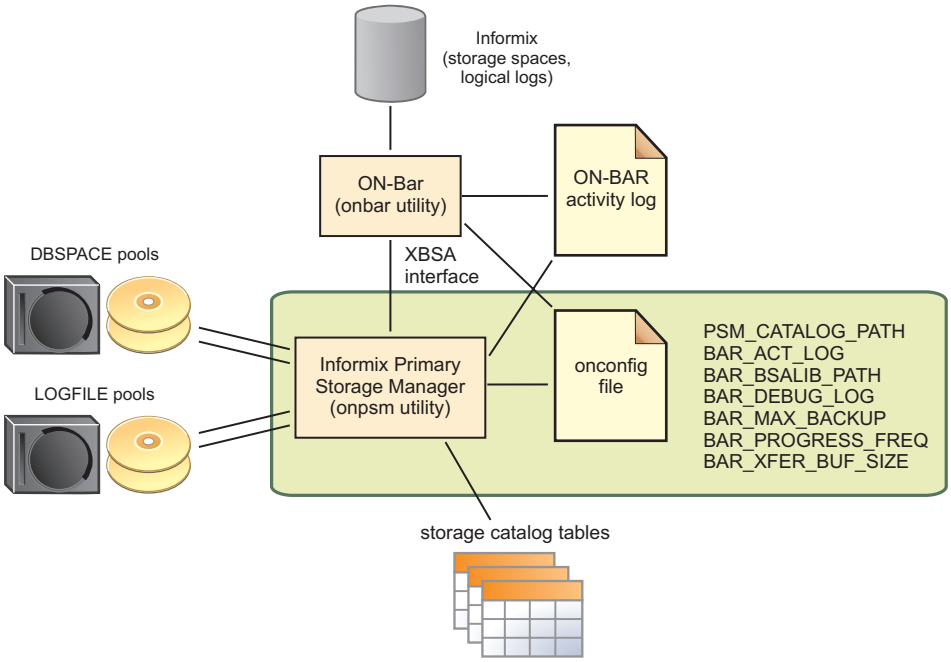


Figure 15-1. Components of Informix Primary Storage Manager

Features of Informix Primary Storage Manager

Storage manager feature	Explanation
Storage devices to use with the storage manager	File devices only The storage manager automatically creates a default device when a catalog is created. The default device is `$INFORMIXDIR/backups`. You can remove the default device.
Buffer transfer size	Unlimited
Encryption and compression	Achieved with BACKUP_FILTER, RESTORE_FILTER FILTERS in ON-Bar (The storage manager does not provide encryption or compression.)
Expiration policies of the storage manager	No expiration policies. (You manually expire backup objects from the storage manager with the **onsmsync** utility. The **onsmsync** object expiration commands remove objects from the storage manager.)

You can perform an imported restore with ON-Bar and the Informix Primary Storage Manager. In an imported restore, you back up an Informix instance on one

machine and restore the instance on a different machine. Use the **onsmsync** export and import options to export the backup objects from the storage manager on the backup machine and import the backup objects into the storage manager on the restore machine.

Related concepts:

"Backup Services API (XBSA)" on page 3-3

"Device pools" on page 15-17

Related tasks:

"Examples: Manage storage devices with Informix Primary Storage Manager"

Related reference:

"Configure a storage manager" on page 4-1

"The onpsm utility for storage management" on page 15-10

"Informix Primary Storage Manager configuration parameters" on page 17-29

Examples: Manage storage devices with Informix Primary Storage Manager

Learn how to set up and use Informix Primary Storage Manager to manage storage devices that the **onbar** utility uses for backing up and restoring instances. Each example shows how you can use the storage manager for a specific backup strategy.

Prerequisites:

- Informix 12.10 is installed with the ON-Bar utility.
- Environment variable INFORMIXDIR is set to the path where the database server is installed.
- Environment variable ONCONFIG is set to the file in `$INFORMIXDIR/etc` that contains the configuration parameters for your database. The name of the file must be unique for each database server instance.
- User **informix** or root privileges.

"Example 1: Storing backups for an instance" on page 15-4

"Example 2: Storing backups for two instances" on page 15-5

"Example 3: Exporting backups to and restoring them from another directory" on page 15-7

"Example 4: Exporting a backup from one server and importing it into another server" on page 15-7

In these examples, *storage manager* refers to Informix Primary Storage Manager.

Related concepts:

Chapter 15, "Informix Primary Storage Manager," on page 15-1

Related reference:

"Informix Primary Storage Manager configuration parameters" on page 17-29

"The onpsm utility for storage management" on page 15-10

"The onsmsync utility" on page 8-4

Example 1: Storing backups for an instance

This example shows how to set up and use Informix Primary Storage Manager to back up the data and logical logs for a single database server instance to a directory: $INFORMIXDIR/backups.

In this example, you update the configuration file so that the Informix Primary Storage Manager can communicate with ON-Bar and you specify the directory where you want backups stored. Then you use the **onbar** utility to perform a standard, level-0 backup of all online storage spaces and used logical logs. You validate the backup by checking the messages that were logged and by using the **onpsm** utility to confirm that storage objects were created.

1. Set the BAR_BSALIB_PATH configuration parameter to the full path and name of the shared library for the storage manager.

   For example, on Linux, Solaris:

   ```
   BAR_BSALIB_PATH $INFORMIXDIR/lib/libbsapsm.so
   ```

   You must use the version of the XBSA shared library that is provided for Informix Primary Storage Manager. If you do not specify the path with the BAR_BSALIB_PATH configuration parameter, you must ensure that the XBSA library is in the default location on your operating system.
2. If needed, create the directory in which to store the backup objects.

   By default, the storage manger includes the default pools LOGPOOL and DBSPOOL, with the default directory $INFORMIXDIR/backups in each pool.

   - If you want to use the default backup directory, verify that the $INFORMIXDIR/backups directory exists.
   - If you want to use a different backup directory, use the **onpsm -D add** command to add a new backup directory for LOGPOOL and DBSPOOL. For example, run the following commands to add different backup directories for the LOGPOOL and DBSPOOL pools:

     ```
     onpsm -D add /backups/infx/logs -g LOGPOOL -p HIGHEST -t FILE
     onpsm -D add /backups/infx/spaces -g DBSPOOL -p HIGHEST -t FILE
     ```

     Use the HIGHEST priority for the device that should be filled first. Only one device in a pool can have the priority setting of HIGHEST.
3. Run the **onbar** utility to perform a standard, level-0 backup of all online storage spaces and used logical logs.

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

   If the storage catalog tables do not exist, they are created in the $INFORMIXDIR/etc/psm directory.
4. Validate that the storage manager is set up and that the backup objects are created.
   a. Look in the ON-Bar activity log to confirm that the storage manager is ready and that ON-Bar recognizes the storage manager.

      For example, the first message is from the storage manager and the second message is from the backup utility:

      ```
      2012-01-03 15:51:23 11193  2569 Informix PSM is ready.
      2012-01-03 15:51:23 11193  2569 Using Informix PSM version 12.10.FC1
      as the Storage Manager. XBSA API version is 1.0.3.
      ```

      By default, the storage manager posts messages to the ON-Bar activity log. The location of the activity log is set by the BAR_ACT_LOG configuration parameter. If you want the storage manager messages to be logged separately, you must set the PSM_ACT_LOG configuration parameter.

b. Run the **onpsm -O list** command to list the storage objects that were created:

   The list, as shown in the following example, includes the storage object IDs, the date the storage objects were created, the size of the storage objects, and where the storage objects are in the storage device. The object IDs are also stored in the `ixbar` file and are used by ON-Bar to locate the objects.

```
============================
Object List Report

                                        Logical Path
Obj ID  Date Created        Size (MB)    Name.Version (omits piece #)
-------
     1  2012-08-06 12:02:10      12.5  /serv1/rootdbs/0/serv1.1
     2  2012-08-06 12:02:12       0.1  /serv1/logdbs/0/serv1.1
     3  2012-08-06 12:02:12       0.1  /serv1/dbs2/0/serv1.1
     4  2012-08-06 12:02:12       0.1  /serv1/dbs1/0/serv1.1
     5  2012-08-06 12:02:13       0.1  /serv1/physdbs/0/serv1.1
     6  2012-08-06 12:02:14       0.3  /serv1/10/9/serv1.1
     7  2012-08-06 12:02:14       0.0  /serv1/crit_files/ixbar/serv1.1
     8  2012-08-06 12:02:14       0.0  /serv1/crit_files/oncfg/serv1.1
     9  2012-08-06 12:02:14       0.1  /serv1/crit_files/onconfig/serv1.1
    10  2012-08-06 12:02:14       0.0  /serv1/crit_files/sqlhosts/serv1.1
============================
```

c. Run the **onpsm -D list** command to display a list that shows that the device was added to the DBSPOOL and LOGPOOL pools. The following example shows output of the command:

```
Type  Prio     Block/Size (MB)  Pool Name   Device Name---
FILE   HIGHEST       --/--         DBSPOOL     /backups/infx/logs

FILE   HIGHEST       --/--         LOGPOOL     /backups/infx/spaces
```

With a few simple steps, you configured the storage manager and performed a full backup of an instance to a file device. Very little configuration was required because the storage manager uses the default settings for various ON-Bar configuration parameters.

Storage catalog tables are not included in an Informix backup. Be sure to back up the storage catalog tables with your operating system tools as part of a disaster recovery strategy. If the storage catalog tables are lost, the **onbar** utility cannot restore the backup objects that Informix Primary Storage Manager created. The location of the storage catalog tables is set by the PSM_CATALOG_PATH configuration parameter (default = `$INFORMIXDIR/etc/psm`).

To restore the instance from the backup objects, use the **onbar** utility. The storage manager tracks the backup objects and storage devices for you.

Example 2: Storing backups for two instances

This example shows how to configure one instance of Informix Primary Storage Manager to manage the storage devices for two database server instances in a multiple residency environment.

In this example, you set up two independent database server environments on the same computer. Each database server is installed in a separate directory: (`/usr/informix/ids1210fc1` and `/usr/informix/ids1210fc1b`) and has a database server instance. Storage for backup operations on both database server instances is managed by one instance of Informix Primary Storage Manager. Pools of storage devices for physical and logical data are configured for each instance.

1. For *each* instance, edit the **onconfig** file to configure storage management for ON-Bar.

Table 15-1. Configuration parameters and their associated values

Configuration parameter	Value
BAR_BSALIB_PATH Specify the full path and name of the shared library for the storage manager.	/usr/informix/ids1210fc1b/lib/libbsapsm.so
PSM_CATALOG_PATH Specify the path of the storage catalog tables.	/usr/informix/ids1210fc1b/etc/psm
PSM_DBS_POOL Specify the name for a group of devices for storing online data (dbspace) backups.	FC1: DBSPOOL_FC1 FC1B: DBSPOOL_FC1B
PSM_LOG_POOL Specify the name for a group of devices for storing online logical log backups.	FC1: LOGPOOL_FC1 FC1B: LOGPOOL_FC1B

2. For *each* instance, create a directory in which to store the backup objects.

```
mkdir $INFORMIXDIR/backups/dev_for_1201fc1
mkdir $INFORMIXDIR/backups/dev_for_1201fc1b
```

3. Run the **onpsm** utility to create device pools for each instance. For example, specify:

```
onpsm -P add DBSPOOL_FC1
onpsm -P add LOGPOOL_FC1
onpsm -P add DBSPOOL_FC1B
onpsm -P add LOGPOOL_FC1B
```

4. Run the **onpsm** utility to add the storage devices.

```
onpsm -D add $INFORMIXDIR/backups/dev_for_1201fc1 -t FILE -g DBSPOOL_FC1
onpsm -D add $INFORMIXDIR/backups/dev_for_1201fc1 -t FILE -g LOGPOOL_FC1
onpsm -D add $INFORMIXDIR/backups/dev_for_1201fc1b -t FILE -g DBSPOOL_FC1B
onpsm -D add $INFORMIXDIR/backups/dev_for_1201fc1b -t FILE -g LOGPOOL_FC1B
```

5. For *each* instance, run the **onbar** utility to perform a standard, level-0 backup of all online storage spaces and used logical logs.

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

6. Validate that the storage manager is set up and that the backup objects are created.
   a. For *each* instance, look in the ON-Bar activity log to confirm that the storage manager is ready and that ON-Bar recognizes the storage manager. For example, look for this information:

```
2012-01-03 15:51:23 11193  2569 Informix PSM is ready.
2012-01-03 15:51:23 11193  2569 Using Informix PSM version 12.10.FC1
as the Storage Manager. XBSA API version is 1.0.3.
```

   b. Use the **onpsm** utility to list the storage objects that were created:

```
onpsm -O list
```

      The report includes the storage object IDs, the date the storage objects were created, the size of the storage objects, and the location of the storage objects in the storage device.

Example 3: Exporting backups to and restoring them from another directory

This example shows how to export backups to a new directory and import the backup objects from that directory.

Suppose that you keep five generations of backups. As an added precaution, you also keep copies of the most recent backups in a separate directory. In this example, you use the **onsmsync** utility to export your most recent backup to and import it from an Informix Primary Storage Manager external pool in a separate directory.

The storage manager tracks devices in the external device pool (EXTPOOL) so it can copy objects to and from external devices. (Although the storage manager tracks devices, it does not track files and objects that are inside the EXTPOOL pool in the storage manager catalogs.)

1. Store backups for an instance, following the steps in "Example 1: Storing backups for an instance" on page 15-4.
2. Run the **onpsm -D list** command to check that there is a device in the EXTPOOL pool.
   a. If there is no device in the EXTPOOL pool, add one using the **onpsm -D add** command.

      The following example shows how to add a device with the path `/export/informix/psm_exportdir` to the EXTPOOL pool.

      ```
      $ onpsm -D add /export/informix/psm_exportdir -g EXTPOOL -t FILE
      ```
3. Run the **onsmsync** command to export all backup objects in the generation 1 level-0 backup, using prefix `pw_sept5`, which becomes the name of the subdirectory in which the utility places the backup:

   ```
   onsmsync -E -p pw_sept5 -g 1
   ```

   After you run the **onsmsync -E** command to export the backup objects, you will see a subdirectory in the EXTPOOL directory that includes a directory holding the backup objects and a file called `export.bom`.

Suppose that something happens to the backup generation stored in your primary backup directory and you want to import the `pw_sept5` backup generation from the second directory. To import the backup generation:

1. Run the **onsmsync** command to import all backup objects in the `pw_sept5` subdirectory:

   ```
   onsmsync -I -p pw_sept5
   ```

Use your own file-transfer methods to move the exported backups, as needed, to other machines.

Example 4: Exporting a backup from one server and importing it into another server

This example shows how to use the **onsmsync** utility to export a backup from a database server that has the name `informix_serv1`. Then the example shows how to use the **onsmsync** utility to import the data into a server that has the name `informix_serv2`.

1. Set up and export files on database server `informix_serv1`:
   a. Set the INFORMIXDIR, INFORMIXSERVER, ONCONFIG, PATH, INFORMIXSQLHOSTS environment variables for `informix_serv1`.

b. Run the **onpsm -D list** command to check that there is a device in the EXTPOOL pool. If there is no device in the EXTPOOL pool, add one using the **onpsm -D add** command.

c. Run the **onsmsync** command to export all backup objects in the generation 1 level-0 backup, using prefix `serv1_20120810`, which becomes the name of the subdirectory in which the utility places the backup:

```
$ onsmsync -E -p serv1_20120810 -g 1
```

2. Prepare to import files on the second database server, `informix_serv2`, as follows:

a. Set the INFORMIXDIR, INFORMIXSERVER, ONCONFIG, PATH, INFORMIXSQLHOSTS environment variables for `informix_serv2`.

b. Run the **onpsm -D list** command to determine if the EXTPOOL has the same device that you viewed or added in step 1b. (This could occur for shared devices). If there is no device in the EXTPOOL pool, add one using the **onpsm -D add** command.

c. Copy the previously exported backup objects (for example, subdirectory `serv1_20120810`) into the EXTPOOL device from which you will import the backup objects.

d. Run the following command to import backup objects from EXTPOOL device:

```
$ onsmsync -I -p serv1_20120810
```

After you run the **onsmsync -I** command to import the backup objects, the objects are stored in the new LOGPOOL and DBSPOOL pools.

e. Run the **onpsm -O list** command to view the imported objects. Notice that the import command also creates a new `ixbar` file in `$INFORMIXDIR/etc/` directory.

```
$ ls -l $INFORMIXDIR/etc/*ixbar*

-rw-rw-- 1 informix informix     0 Aug 10 19:44
 /usr/informix/etc/ixbar.12.20120810.194441
-rw-rw-- 1 informix informix  2704 Aug 10 19:44
 /usr/informix/etc/ixbar.12
```

The new `ixbar` file lists the imported backup objects so that you can perform an ON-BAR cold restore to restore the `informix_serv1` instance from the first database server to the `informix_serv2` instance on the second database server.

Setting up Informix Primary Storage Manager

Setting up involves gathering and specifying information about your storage devices and, if necessary, changing the default configuration of the storage manager.

Related reference:

"Configure a storage manager" on page 4-1

"The onpsm utility for storage management" on page 15-10

"Informix Primary Storage Manager configuration parameters" on page 17-29

Collecting information about file directories and devices

You must gather information about and configure at least one file directory or device for each of the DBSPOOL and LOGPOOL pools before ON-Bar can use the IBM Informix Primary Storage Manager.

Before defining directories or devices, gather the following information:

- The full path names and types of the devices that you plan to use for your backup storage.
- The amount of space that you want to commit to ON-Bar backups.

Configuring Informix Primary Storage Manager

By default, the IBM Informix Primary Storage Manager is automatically configured with the information specified in the storage manager and with some ON-Bar configuration parameters. It is also automatically configured when you use the **onpsm** utility. You can change the configuration.

The Informix Primary Storage Manager uses file devices (disks) only, not tapes. You cannot configure the storage manager to use tapes.

To manually configure the Informix Primary Storage Manager:

1. Update the BAR_BSALIB_PATH configuration parameter to point to the storage manager library.

   For example, on Linux or Solaris, specify:

   ```
   BAR_BSALIB_PATH $INFORMIXDIR/lib/libbsapsm.so
   ```

2. Specify the destination and source devices for backup and restore operations by using the **onpsm** utility.
3. Change the default configuration for the storage manager if necessary for your environment:
   a. To override the default values for the location of storage manager log files and catalogs, debugging activity, and pool names, specify new values in the Informix Primary Storage Manager configuration parameters.
   b. To specify a larger transfer buffer with ON-Bar and the Informix Primary Storage Manager, increase the size in the BAR_XFER_BUF_SIZE configuration parameter.
   c. To change the frequency of the progress messages in the ON-Bar activity log, update the value specified in the BAR_PROGRESS_FREQ configuration parameter.
   d. To change the number of processes that ON-Bar runs concurrently, update the value specified in the BAR_MAX_BACKUP configuration parameter.

Related reference:

"Informix Primary Storage Manager configuration parameters" on page 17-29

"ON-Bar and ontape configuration parameters and environment variable" on page 17-1

"The onpsm utility for storage management" on page 15-10

Managing storage devices

Use the **onpsm** utility to add, monitor, and remove storage devices and to manage IBM Informix Primary Storage Manager catalogs, locks, and objects. Use the **onsmsync** utility to export ON-Bar backups to and import them from external pools and to expire backups.

Related reference:

"The onpsm utility for storage management" on page 15-10

"The onsmsync utility" on page 8-4

The onpsm utility for storage management

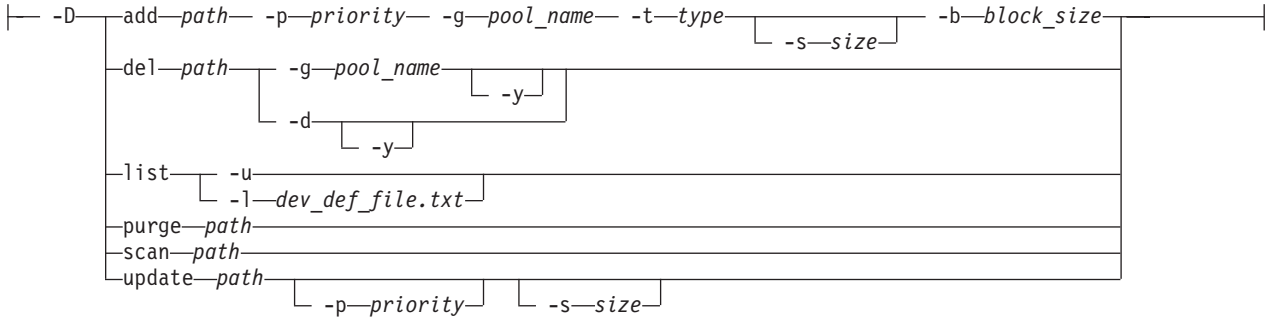
Use the **onpsm** utility to manage the IBM Informix Primary Storage Manager catalogs, devices, locks, and objects.

Pre-requisite: To run the **onpsm** utility, you must be user **root** or **informix** or a member of the **bargroup** group.

Syntax

Catalog options:

Device options:



Object options:

Pool options:

Table 15-2. **onpsm** *utility catalog options*

Element	Purpose	Key Considerations
- C check	Checks storage manager catalog tables, which store metadata about the pools and devices that the storage manager manages	This command identifies files that have problems.
- C check -l	Displays index keys while checking the catalog tables	
- C check -n	Indicates that the storage manager does not fix errors that are found	
- C detail	Shows details about the storage manager catalog tables	
- C export	Exports the Informix Primary Storage Manager catalog tables to a directory called `psm_catalog.exp`	
- C import	Replaces the current Informix Primary Storage Manager catalog with a catalog that is recreated from files that are in the `psm_catalog.exp` directory	Only import the catalog if you have a system problem, lost your current catalog, and need to revert to the exported catalog. If you need to import a catalog, run the **onpsm -C init** command before you run the **onpsm -C import** command.
- C init	Deletes storage manager catalog tables	
- C init -d	Deletes storage manager catalog table and the backup objects in file devices	
- C unlock	Unlocks the storage manager catalog	If the storage manager exits abnormally from a backup or restore session because a failure occurred, storage manager catalog tables might remain locked. If catalog tables are locked, you can release the locks.
-y	Specifies to not to ask for confirmation before deleting catalog tables	

Table 15-3. **onpsm** *utility device options*

Element	Purpose	Key Considerations
-D add	Adds a device to the pool specified with the **-g** option	Before adding devices, gather information about the device. See "Collecting information about file directories and devices" on page 15-8.

Table 15-3. **onpsm** *utility device options (continued)*

Element	Purpose	Key Considerations
-D del	Removes a device: • If you use the **-g** option, removes a device from the pool specified with the **-g** option, while retaining the device objects in the Informix Primary Storage Manager catalog. • If you use the **-d** option, removes the device from all pools and removes all backup objects from the file system in that device	If you delete the device using **-g** option, you can restore the objects if necessary. If you remove a device, the storage manager cannot add new objects to the device.
-D list	Displays a list of all devices in the system	
-D purge	Removes missing storage manager objects from the Informix Primary Storage Manager catalog	
-D scan	Scans objects in the device to verify that the objects exist in the Informix Primary Storage Manager catalog so the objects can be restored if necessary If an object is not in the catalog, this command adds the object to the catalog.	If the command cannot add an object to the catalog, the command ignores the missing file. Before a missing object can be added to a catalog, the following conditions must occur: • The object ID must not be assigned to any other object in the storage manager. • Files must not be renamed or relocated to different directories inside the device • The object version must not be assigned to any other object in the storage manager.
-D update	Modifies information about a device	If you want to modify information about more than one device, run a separate command for each device.
path	Full name and path to the device (for TAPE devices) or to a directory (for FILE devices)	The path must be in the format appropriate to the operating system to which the device is attached . The name of the device must be unique within a pool. You can include the same device in multiple pools. If you are deleting, listing, purging, scanning, or updating information, the path must be to an existing device.
-b ***block_size***	For tape devices only, the minimum number of bytes of data that need to accrue before the data is written to the device	The block size is required for tape devices.
-d	Deletes the pool from all pools and deletes backup objects	The block size is required for tape devices.

Table 15-3. **onpsm** *utility device options (continued)*

Element	Purpose	Key Considerations
-g *pool_name*	The pool in which to add the device, either DBSPOOL, LOG POOL, or EXTPOOL	Information about pools is stored in the Informix Primary Storage Manager catalog. If you do not provide a pool name, the command fails. Specify: • DBSPOOL for backups of dbspaces, blobspaces, and sbspaces • LOGPOOL for backups of logical logs • EXTPOOL that serves as a staging area from which you can move specific backups or backup generations to permanent storage or onto a different computer.
-l *dev_def_file.txt*	Loads information about the device from a device-definition file	
-p ***priority***	Priority of the device, either HIGHEST, HIGH, LOW, or READ-ONLY	The storage manager fills high-priority devices in a pool before placing data into low-priority devices in that pool. If the high-priority devices are busy at the moment when the storage manager is ready to fill a pool, the storage manager uses low priority devices. Only one device in a pool can have the priority of HIGHEST. If multiple devices have the same priority in the same pool, the storage manager determines which device to use first. When a device becomes full, the storage manager changes the priority to READ-ONLY. You can change the priority after you add more space to the device.
-s ***size***	For tape devices only, the maximum storage capacity of the device in kilobytes	The size is optional for tape devices. If a size is not specified, or if you specify `0`, the storage manager interprets the size as unlimited. When the size is unlimited, the device is not considered full until it returns an error that specifies that the device is full. To specify the size enter the numeric value of the size followed by the suffix B, K, M, G, T, or P (for bytes, kilobytes, megabytes, gigabytes, terabytes, or petabytes). The suffix can be upper or lowercase.
- t ***type***	Type of device, either FILE or TAPE	Information about devices is stored in the Informix Primary Storage Manager catalog,
-u	Unloads information about the device to a device-definition file	The device definition file is a text file with a specific format. The storage manager uses the file to recreate the information when you run an **onpsm** command with the load option.
-y	Specifies not to ask for confirmation to complete the requested action	

Table 15-4. **onpsm** *object options*

Element	Purpose	Key Considerations
-O del	Deletes physical objects from a pool	

Table 15-4. **onpsm** *object options (continued)*

Element	Purpose	Key Considerations
-O detail	Displays details about the specified object. Details include the location of the object.	
-O dump	Extracts the object data to a file in the current directory	
- o ***object_id***	Identifies the particular object	You can delete or dump one or more objects with a single command, as shown in "Usage."
-O list	Displays all objects in a pool	For each object, the list includes the date and time the object was created, the size of the object, and the path name of the object.
-y	Specifies not to ask for confirmation to complete the requested action.	

Table 15-5. **onpsm** *pool options*

Element	Purpose	Key Considerations
-P add *pool_name*	Adds a new pool	
P del *pool_name*	Deletes the specified pool	
-P list	Lists all pools in the system	
-y	Specifies not to ask for confirmation to complete the requested action.	

Table 15-6. **onpsm** *utility general options*

Element	Purpose	Key Considerations
-h	Displays help information	
-V	Displays the software version number and the serial number	For more details about the standard Informix **-V** and **-version** options, see Obtaining utility version information.
-version	Displays the software version number, serial number, and additional information such as the host, operating system, build date, and the Global Language Support (GLS) version	For more details about the standard Informix **-V** and **-version** options, see Obtaining utility version information.
-version all	Displays **onpsm** version information and information about the PSM shared library	

Usage

When you run an **onpsm** command to define a device, the storage manager automatically creates storage manager catalogs if they do not exist.

The default device for the storage manager is `$INFORMIXDIR/backups`. The device, which is low priority, is automatically created when the catalog is created. You can remove the default device.

When you create a device, the storage manager automatically creates the directory for the device if the directory does not exist. The storage manager uses the directory path that you specify in the **onpsm -D add** command.

You can delete one or more objects with a single command, for example, by running a command that has this format:

```
onpsm _O del -o obj_1 -o obj_2
```

You can also dump one or more objects with a single command, for example, by running a command that has this format:

```
onpsm _O dump -o obj_1 -o obj_2
```

If the data is not needed, run the **onsmsync** utility to delete backup objects from the Informix Primary Storage Manager.

Some third-party storage managers do not allow the **onsmsync** utility to delete backup objects from the storage manager. If you have a third-party storage manager, you might need to manually delete backup objects that you no longer need.

Examples

The following command adds a file device with the path name `$INFOMRIXDIR/backups` in the DBSPOOL pool:

```
onpsm -D add $INFORMIXDIR/backups -g DBSPOOL -t FILE  -p HIGH
```

The following command checks Informix Primary Storage Manager catalog tables and indicates that the storage manager does not fix any errors found during the check:

```
onpsm -C check -n
```

The following command lists objects in pools, including the date and time the object was created, the size of the object, and the path name of the object.

```
onpsm -O list
```

Related concepts:

Chapter 15, “Informix Primary Storage Manager,” on page 15-1

“Setting up Informix Primary Storage Manager” on page 15-8

“Managing storage devices” on page 15-9

“Device pools” on page 15-17

Related tasks:

“Configuring Informix Primary Storage Manager” on page 15-9

“Examples: Manage storage devices with Informix Primary Storage Manager” on page 15-3

onpsm -C detail output

Use the **onpsm -C detail** command to view details about the storage manager catalog tables.

Sample onpsm -C detail command output

```
D:\IFMXDATA\gacpsm>onpsm -C detail

Informix Primary Storage Manager State:

    PSM Unique ID    : 1358735848
    Catalog Location : D:\Informix\ids1210\etc\psm\
    Catalog State    : Locked
    Catalog Owner    : 2
    Catalog Lock Mode: Regular
```

```
Sessions:
    Session ID       Process ID
    2                1576
    3                4556
    4                5555

Informix PSM Locked Objects

Session  Object Id Date       Time     Server        Object Name

     3         116 2012-12-09 20:54:34 /gacpsm_tcp   /gacpsm_tcp/168/242
     4         117 2012-12-09 21:07:44 /gacpsm_tcp   /gacpsm_tcp/168/242
```

The output contains the following sections:

Informix Primary Storage Manager State
Shows general information about all of the sessions that are active in the system and whether the catalog is locked.

PSM Unique ID
ID for the catalog

Catalog Location
Path to the catalog

Catalog State
Indicates whether the catalog is locked

Catalog Owner
Informix Primary Storage Manager session ID

Catalog Lock Mode
Lock category

For example, `Regular` means that the lock is a user lock.

Sessions
Lists all of the sessions that are active in the system and the process IDs that match the sessions.

Session ID
ID of the session. This is the same ID that appears in the `Catalog Owner` field.

Process ID
Internal ID for the ON-Bar, **archecker**, or storage manager process that is locking the catalog.

Informix PSM Locked Objects
Shows the locks in devices or objects that are held by storage manager sessions. For each lock, the output shows the session number, the object ID, the date and time that the lock was placed, the server, and the object name.

onpsm -D list output

The **onpsm -D list** command displays information about all of the devices in each IBM Informix Primary Storage Manager pool. You can use this list to determine whether you need to change information about your devices.

Sample onpsm -D list command output

```
Type   Prio     Block/Size (MB)  Pool Name     Device Name
-----  -------  ---------------  ------------  ---------------------------
FILE    LOW         --/--        DBSPOOL      /informix/backups

FILE    LOW         --/--        LOGPOOL      /informix/backups
```

Type Type of device, either FILE or TAPE (Currently only the FILE type is supported.)

Prio Priority of the device, either HIGH, HIGHEST, LOW, or READ-ONLY

HIGH is the default priority if a priority is not specified. Only one device in a pool can have a priority of HIGHEST.

Block Size
Size of the device (only applies to devices of the TAPE type)

Pool Name
The name of the pool (DBSPOOL, LOGPOOL, or EXTPOOL)

In the example output above, there are no EXTPOOL devices.

Device Name
Complete path name for the device

onpsm -O list output

The **onpsm -O list** command displays all of the objects stored in a pool.

Sample onpsm -O list command output

```
Object List Report
                                        Logical Path
Obj ID  Date Created         Size (MB)    Name.Version (omits piece #)
------  -------------------  ---------  ----------------------------------
     1  2012-07-06 14:39:47       12.0  /gacpsm_tcp/rootdbs/0/gacpsm_tcp.1
     2  2012-07-06 14:41:18       12.0  /gacpsm_tcp/rootdbs/0/gacpsm_tcp.2
     3  2012-07-11 13:42:10        3.9  /gacpsm_tcp/160/14/gacpsm_tcp.1
     4  2012-07-11 13:42:13        3.9  /gacpsm_tcp/160/15/gacpsm_tcp.1
     5  2012-07-11 13:42:15        3.9  /gacpsm_tcp/160/16/gacpsm_tcp.1
     6  2012-07-11 13:42:15        3.9  /gacpsm_tcp/160/17/gacpsm_tcp.1
     7  2012-07-11 13:42:15        3.9  /gacpsm_tcp/160/18/gacpsm_tcp.1
     8  2012-07-11 13:42:15        3.9  /gacpsm_tcp/160/19/gacpsm_tcp.1
     9  2012-07-11 13:42:16        3.9  /gacpsm_tcp/160/20/gacpsm_tcp.1
    10  2012-07-11 13:42:16        3.9  /gacpsm_tcp/160/21/gacpsm_tcp.1
```

Obj ID
The ID of the stored object

Date Created
The date and time when the object was created

Size The size of the object in megabytes

Name.Version
The path name of the object followed by `.` and the version of the object (for example, `.2` to indication version 2)

Device pools

An IBM Informix Primary Storage Manager pool is a named group of disk devices that you use as a repository for backups.

When storing backup objects, the Informix Primary Storage Manager selects particular devices from the pool and moves automatically from one device to another when devices are full or when they fail. You maintain pools by using the **onpsm** utility to add, modify, view, and drop devices in the pool.

Three pools are available:

DBSPOOL
Holds online backups of dbspaces, blobspaces, and sbspaces

LOGPOOL
Holds online backups of logical logs

EXTPOOL
Serves as a staging area for exporting a backup set of objects to a single large, external logical object or for importing the backed up objects. You can move specific backups or backups generations in this pool to permanent storage or onto a different computer. Files in the EXTERNAL pool are offline. The files are not visible to ON-Bar, and the Informix Primary Storage Manager does not track them.

Related concepts:

Chapter 15, "Informix Primary Storage Manager," on page 15-1

"Informix Primary Storage Manager file-naming conventions" on page 15-19

Related reference:

"PSM_DBS_POOL configuration parameter" on page 17-31

"PSM_LOG_POOL configuration parameter" on page 17-34

"The onpsm utility for storage management" on page 15-10

Device-configuration file for the Informix Primary Storage Manager

The **onpsm** utility can generate a device-configuration file, which is a text file that contains information about a storage device. The utility uses this information to re-create the devices.

The configuration file contains the following information about each device:

DEVICE
The complete path to the device

TYPE Type of device
FILE = file directory on a disk device

POOL The pool that contains the device, either DBSPOOL, LOGPOOL, or EXTPOOL

BLOCKSIZE
Not applicable for disk devices.

SIZE Not applicable for disk devices.

PRIORITY
The priority of the device

For example, the device-configuration file might contain the following information

```
DEVICE=/vobs/tristarm/sqldist/psm_backup/dbspace
TYPE=FILE
POOL=DBSPOOL
BLOCKSIZE=0
SIZE=0
PRIORITY=HIGH
```

Informix Primary Storage Manager file-naming conventions

When creating files that store your backup data, the IBM Informix Primary Storage Manager uses specific file-naming conventions.

For the DBSPOOL and LOGPOOL device pools, the path name of the storage file consists of:

1. Category information:
   - For space backups, the category consists of the device name, the database server name, and the space name
   - For log backups, the category consists of the device name, the database server name, server number, and a log file number
2. For space backups, the backup level, either: 0, 1, or 2
3. A version number, which is an integer that starts at 1 for an object of that category and is incremented for each subsequent backup of an object of that category.
4. An ID that identifies a piece of the backed up object

For space backups, file names have this format:

/device/DBSERVERNAME/dbspace/backup_level/DBSERVERNAME.version.piece

For example, the name of the file for the third piece of a second level 0 backup of the dbspace `rootdbs` for the server named `SERVER1` is:

```
/my_device/SERVER1/rootdbs/0/SERVER1.2.3
```

For log backups, file names have this format:

/device/DBSERVERNAME/SERVERNUM/LOG_UNIQUE_ID/DBSERVERNAME.version.piece

If you use the **onsmsync** utility with the **-E** option to export a backup generation in the Informix Primary Storage Manager, the **onsmsync** utility creates and places the backup files in a subdirectory of the storage manager EXTPOOL device. You must provide a *prefix* (the name of a subdirectory) when you use the **onsmsync** utility with the **-E** or **-I** options. ON-Bar uses the specified path as a key to communication with the storage manager to store and retrieve objects.

Related concepts:

"Device pools" on page 15-17

Related reference:

"The onsmsync utility" on page 8-4

Message logs for Informix Primary Storage Manager

The Informix Primary Storage Manager writes messages to a storage manager activity log and debug log.

The message logs are stored in the directories specified in the BAR_DEBUG_LOG or BAR_ACT_LOG configuration parameters. You can use the PSM_ACT_LOG and PSM_DEBUG_LOG configuration parameters to specify another directory for each of these logs.

The PSM_DEBUG configuration parameter specifies the level of debugging activity that is captured in the debug log.

Related reference:

"PSM_ACT_LOG configuration parameter" on page 17-29

"PSM_DEBUG_LOG configuration parameter" on page 17-33

"PSM_DEBUG configuration parameter" on page 17-32

Part 5. archecker table level restore utility

Chapter 16. archecker table level restore utility

You can use the **archecker** utility to perform point-in-time table-level restores that extract tables or portion of tables from archives and logical logs.

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.

For information about using the **archecker** utility to verify backups, see "onbar -v syntax: Verifying backups" on page 5-12.

Related reference:

"The archecker utility configuration parameters and environment variable" on page 17-24

Overview of the archecker utility

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.

The **archecker** utility is not JSON compatible. If you try to use the utility with target tables that contain JSON or BSON (binary JSON) data types, the utility will abort and return an error message.

Use **archecker** utility 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.

  You cannot restore data from a remote device.

  You cannot use a shared memory connection when performing a table-level restore.

- 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. Other situations require that you use different utilities. For example, 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.

Related reference:

"ON-Bar script" on page 3-5

The archecker 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 file name.

For information about the configuration parameters used in this file, see "The archecker utility configuration parameters and environment variable" on page 17-24.

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 about the supported statements and syntax, see "The archecker schema reference" on page 16-8.

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 configuration parameter" on page 17-27.
- Use the `-f` *cmdname* command-line option. For more information, see "Syntax for archecker utility commands" on page 16-5.

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 is used for all of the tables being restored by using the same table-level restore command schema file.

Data restore with archecker

Use the **archecker** utility to perform to types of restore operations.

The two types of restores that the **archecker** utility performs are:

- A physical restore that 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

When the **archecker** utility performs a physical restore, the utility extracts data 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.

To restore a table with the original schema, the source schema must be specified. To restore a table with a different schema, the table name in the target schema must be different from the table name in the source schema. After restoring by using a different schema, the table can be renamed with the **rename table** statement.

Logical restore

After a 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.

The **archecker** utility cannot process compression dictionaries during a logical restore of compressed tables in non-logged databases. A logical restore stops for a table if it finds that a new compression dictionary was created for that table.

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 in to 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.

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 16-13). 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.

Syntax for archecker utility commands

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.

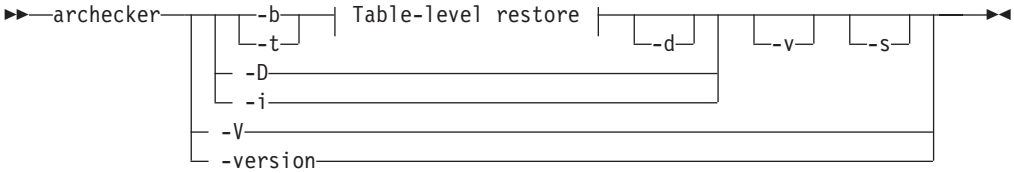


Table-level restore:

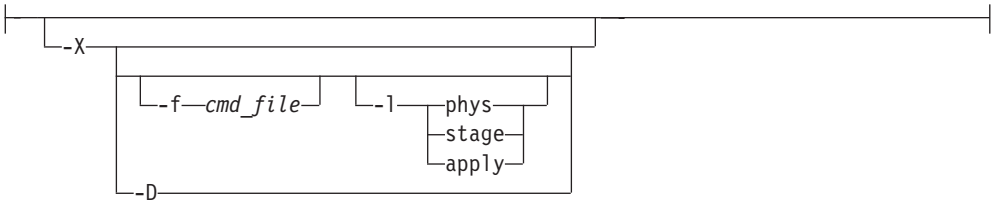


Table 16-1. Options for the archecker command

Element	Description
-b	Provides direct XBSA access for backups created with ON-Bar.
-d	Deletes previous **archecker** restore files, except the **archecker** message log. For more information, see "When to delete restore files" on page 16-8.
-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 16-8.
-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 16-2.
-i	Manually initializes the system.

Table 16-1. Options for the archecker command (continued)

Element	Description
-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 control a logical restore."
-s	Prints a status message to the screen.
-t	Specifies **ontape** as the backup utility.
-v	Specifies verbose mode.
-X	Specifies a table-level restore.
-V	Displays IBM Informix version information.
-version	Displays additional version information about the build operation system, build number, and build date for IBM Informix.

When you use ON-Bar, you can use an ON-Bar command to access **archecker** information to verify a backup. For information on the syntax for this command, see "onbar -v syntax: Verifying backups" on page 5-12.

Manually control a logical restore

You can manually control the stager and applier with the `-l` command-line option.

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

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 **archecker -DX** command on a single node.
4. Start the physical restore by executing the **archecker -bX -lphys** command on each node.

   Restriction: Do not use the -d option.
5. After the physical restore completes, start the logical restore by executing the **archecker -bX -lstage** command on each node that contains logical log records.

   Restriction: Do not use the -d option.
6. After starting all stagers, complete the restore by executing the **archecker -bX -lapply** command on a single node.

Perform 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.

Restore tables with large objects

ON-Bar supports table-level restores of smart large objects and binary large objects.

- Smart large objects

  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.
- Binary large objects

  Table-level restore supports restoring tblspace binary large objects, but not blobspace binary large objects. If you attempt to restore a blobspace binary large object, the value is set to NULL and a warning is issued.

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

The topics in this section describe the SQL-like statements used by the **archecker** schema command file. This file provides information that the **archecker** utility uses to perform data recovery.

Use the schema command file to specify the source and destination tables and to define the table schema.

For information about specifying which command file **archecker** uses, see "Schema command file" on page 16-2.

The following are statements supported by **archecker**:

- CREATE TABLE
- DATABASE
- INSERT INTO
- RESTORE
- SET

Important: Standard SQL comments are allowed in the **archecker** utility file and are ignored during processing.

The syntax of these statements is described in the following topics.

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 16-10.

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.

The schema of the source table is not validated by **archecker**. Failing to provide an accurate schema leads 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 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:

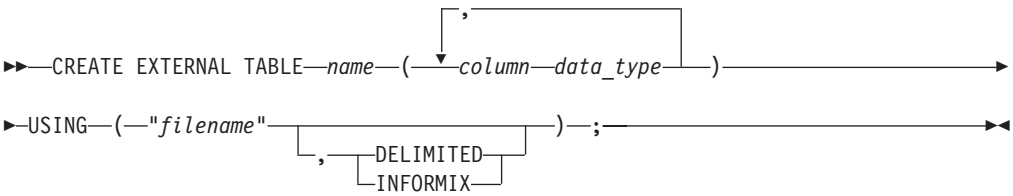
```
CREATE TABLE source (col1 integer, ...) IN dbspace1;
CREATE TABLE target (col1 integer, ...)
   FRAGMENT BY EXPRESSION
      MOD(col1, 3) = 0 in dbspace3,
      MOD(col1, 3) = 1 in dbspace4,
      MOD(col1, 3) = 2 in dbspace5;
INSERT INTO target SELECT * FROM source;
```

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*.
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 before 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:

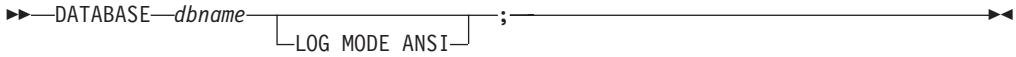
- DELIMITED: ASCII delimited file. This is the default format.
- INFORMIX: internal binary representation. To optimize performance, filters are not applied to external tables. If filters exist, a warning indicates that they are ignored.

For an example of using the CREATE EXTERNAL TABLE statement, see "Restore to an external table" on page 16-15.

The DATABASE statement

In the **archecker** utility, 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.

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;
```

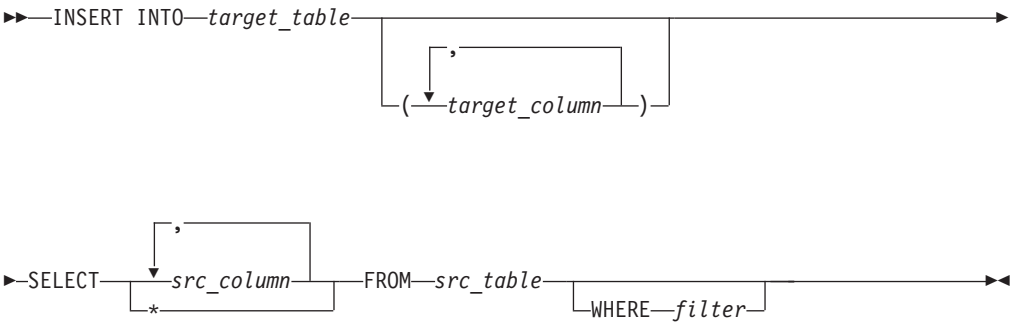
You can use multiple database statements to extract a table from one database into another database.

```
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] MATCHES, [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;
```

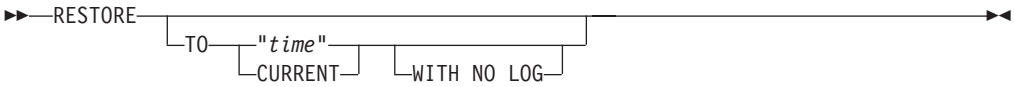
You can also extract a subset of columns. In the following example, only two columns from the source table are inserted into the destination table.

```
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

The RESTORE statement is an optional command to restore tables to a specific point in time.

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.

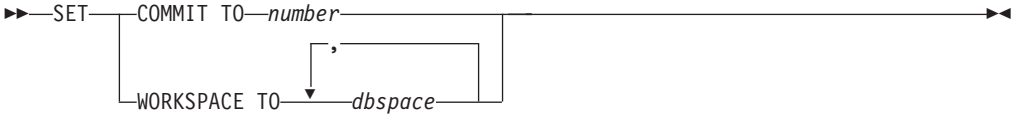
Example

```
RESTORE TO CURRENT WITH NO LOG;
```

The SET statement

The SET statement controls the different features in the table-level unload library.

Syntax



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.

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;
```

Schema command file examples

This section contains examples that 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;
```

Restore 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';
```

Restore 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;
```

Extract 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;
```

Use 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.

Important: 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;
```

Restore 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;
```

Restore 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;
```

Perform 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;
```

Part 6. Backup and restore configuration parameter reference

Chapter 17. Backup and restore configuration parameters

These topics describe the configuration parameters that you use with the ON-Bar, **ontape**, and **archecker** utilities.

You set most of these configuration parameters in the `onconfig` file. However, you set some of the **archecker** configuration parameters 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 4-6, 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.
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

Related reference:
Part 2, "ON-Bar backup and restore system"

ON-Bar and ontape configuration parameters and environment variable

Many properties of the ON-Bar and **ontape** utilities are controlled by configuration parameters in the `onconfig` file. ON-Bar also has an environment variable.

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 `NUL` on Windows.

Related tasks:

"Configuring Informix Primary Storage Manager" on page 15-9

BACKUP_FILTER configuration parameter

Use the BACKUP_FILTER configuration parameter to specify the path name and any options for an external filter program that you use with the ON-Bar or **ontape** utility.

onconfig.std value
: Not set. Backup data is not filtered.

values
: The path name of a command and any options. By default, the path name is relative to the `$INFORMIXDIR/bin` directory, otherwise, the path name must be the absolute path of the program. If you include command-line options, both the filter name and the options must be surrounded by single quotation marks.

takes effect
: After you edit your `onconfig` file and ON-Bar or **ontape** starts.

Usage

This filter transforms data before backing it up, such as compressing it. The transformed data is then backed up and stored as a single file. When you perform a restore, you must transform the data back to its original format. Specify the appropriate program to transform data before a restore by setting the RESTORE_FILTER configuration parameter.

For security purposes, filters should not have write permission to non-privileged users. Permission on the filters is the same as that of permission on other executable files that are called by the IBM Informix server or utilities.

Note: If the BACKUP_FILTER parameter is set in the `onconfig` file, the LTAPESIZE configuration parameter cannot be set to 0. Otherwise the ON-Bar or **ontape** utility returns an error when backing up logical logs to a directory on disk. The error message is:

```
The LTAPESIZE configuration parameter cannot be set to 0 when the BACKUP_FILTER
configuration parameter is set; change the value of LTAPESIZE.
Program over.
```

A workaround is to set the LTAPESIZE configuration parameter to a high value. Log files are not much higher than the LOGSIZE configuration parameter. Use the value in the LOGSIZE as the upper limit for this database.

When you specify filter information in the BACKUP_FILTER configuration parameter, specify the path name of a filter program, and any options, as shown in this example:

```
BACKUP_FILTER  /bin/compress
```

Output produced by this filter is 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. For example, specify:

```
BACKUP_FILTER 'my_encrypt -file /var/adm/encryption.pass'
```

Related reference:

"RESTORE_FILTER configuration parameter" on page 17-20

BAR_ACT_LOG configuration parameter

Use the BAR_ACT_LOG configuration parameter to specify the full path name of the ON-Bar activity log.

onconfig.std value

UNIX: BAR_ACT_LOG $INFORMIXDIR/tmp/bar_act.log

Windows: BAR_ACT_LOG %INFORMIXDIR%\tmp\bar_act.log

range of values

Full path name

takes effect

When **onbar-driver** starts

When you reset the value dynamically in your `onconfig` file by running the **onmode -wf** or equivalent SQL administration API command.

Usage

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.

The file specified by the BAR_ACT_LOG configuration parameter is created if it does not exist. If the ON-Bar command (or any ON-Bar-related utility such as the **onsmsync** utility) never ran on the system, then the file does not exist.

The **sysbaract_log** table is a system monitoring interface pseudo table that reads data from the file specified by BAR_ACT_LOG. The following errors are returned if you attempt to query the **sysbaract_log** on a system where the BAR_ACT_LOG file does not exist:

```
244: Could not do a physical-order read to fetch next row.
101: ISAM error:  file is not open.
```

Usage when you specify a file name only

If you specify a file name only in the BAR_ACT_LOG configuration 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.

For UNIX, 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.

For Windows, if the database server launches a continuous logical-log backup, ON-Bar writes to the activity log in the `%INFORMIXDIR%\bin` directory instead.

Related concepts:

"bar_act.log file: ON-Bar activity log" on page 3-4

Related reference:

"View ON-Bar backup and restore performance statistics" on page 8-10

onmode -wf, -wm: Dynamically change certain configuration parameters (Administrator's Reference)

"PSM_ACT_LOG configuration parameter" on page 17-29

BAR_BSALIB_PATH configuration parameter

Use the BAR_BSALIB_PATH configuration parameter to specify the path name and file name of the XBSA shared library for the storage manager that you use.

default value

UNIX: $INFORMIXDIR/lib/ibsad001.*extension*

Windows: %INFORMIXDIR%\lib\ibsad001.*extension*

The *extension* is platform specific.

onconfig.std value

Not set.

takes effect

When **onbar-driver** starts

When you reset the value dynamically in your `onconfig` file by running the **onmode -wf** or equivalent SQL administration API command.

Usage

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 whether you can use this configuration parameter 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:

UNIX:

- Place the storage-manager library in the default directory.

  For example, the suffix for Solaris is `so`, so you specify `$INFORMIXDIR/lib/libbsapsm.so` on a Solaris system.
- Place the storage-manager library in any directory and create a symbolic link from `$INFORMIXDIR/lib/ibsad001.`*`platform_extension`* to it.
- Set the **LD_LIBRARY_PATH** environment variable. For example, set **LD_LIBRARY_PATH** to `$INFORMIXDIR/lib`.

Windows:

- Place the storage-manager library in the default directory.

If the parameter BAR_BSALIB_PATH is missing or has no value and the database server cannot open the XBSA shared library for your platform, ON-BAR tries to use the IBM Informix Primary Storage Manager as the storage manager in all platforms.

Tip: Be sure that the shared library can access the backup data in the storage manager in a restore. You cannot back up on to one storage manager and restore from a different storage manager.

Related reference:

onmode -wf, -wm: Dynamically change certain configuration parameters (Administrator's Reference)

BAR_CKPTSEC_TIMEOUT configuration parameter

The BAR_CKPTSEC_TIMEOUT configuration parameter specifies the amount of time, in seconds, that an RS secondary server should wait for a checkpoint to arrive from the primary server while performing an external backup.

onconfig.std value

UNIX: 15

Windows: 16

default value

UNIX: 15

Windows: 16

units seconds

range of values

5 through twice the value of the CKPTINTVL configuration parameter

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the **onmode -wf** command.

When you reset the value in memory by running the **onmode -wm** command.

Usage

When an external backup is performed on an RS secondary server, the secondary server must wait until a checkpoint arrives in the logical logs from the primary server. A checkpoint flushes buffers to disk, and blocks user transactions that involve temporary tables. If the checkpoint on the primary does not complete in the time-out period, the backup on the RS secondary server fails. You can set the BAR_CKPTSEC_TIMEOUT configuration parameter to a longer amount of time, in seconds, that an RS secondary server should wait for a checkpoint to arrive from the primary server while performing an external backup.

Related reference:

onmode -wf, -wm: Dynamically change certain configuration parameters (Administrator's Reference)

CKPTINTVL configuration parameter (Administrator's Reference)

BAR_DEBUG configuration parameter

Use the BAR_DEBUG configuration parameter to specify the amount of debugging information that the database server captures in the ON-Bar activity log.

onconfig.std value

BAR_DEBUG 0

values 0 = Do not display debugging information.

1 = Print a small amount of information

2 = Print a message every time ON-Bar:

- Enters a function.
- Exits a function. The message includes the return code for the function.

3 = Print exit and entry information with additional details.

4 = Also print information about ON-Bar parallel operations.

5 = Also print information about:

- Objects that are being backed up or restored.
- The act_node structure corresponding with the bar_action table.

6 = Print additional information about:

- Objects that are being backed up or restored.
- The act_node structure corresponding with the bar_action table.

7 = Also print:

- Information about the contents of the ins_node structure corresponding with the bar_instance table.
- Information about modifications to the bar_action table.
- Information about logical logs and objects that are restored.
- SQL statements done on the **sysutils** database and SQLCODES that were returned.

8 = Also print page headers of all pages archived and restored. This setting requires a large amount of space.

9 = Print the contents of:

- The bar_ins structure after it was initialized.
- The object descriptors that are cold restored.

takes effect

Immediately after you edit your `onconfig` file 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.

When you reset the value dynamically in your `onconfig` file by running the **onmode -wf** or equivalent SQL administration API command.

Usage

The default value of 0 displays no debugging information. Set the BAR_DEBUG configuration parameter to a higher value to display more detailed debugging information in the ON-Bar activity log.

You can dynamically update the value of BAR_DEBUG in the `onconfig` file during a session.

Related reference:

"BAR_DEBUG_LOG configuration parameter" on page 17-7

onmode -wf, -wm: Dynamically change certain configuration parameters (Administrator's Reference)

BAR_DEBUG_LOG configuration parameter

Use the BAR_DEBUG_LOG parameter to specify the location and name of the ON-Bar debug log.

onconfig.std *value*
: `/usr/informix/bar_dbug.log`

if value not present
: UNIX: `/tmp/bar_dbug.log`

  Windows: `\tmp\bar_dbug.log`

takes effect
: When ON-Bar starts

  When you reset the value dynamically in your `onconfig` file by running the **onmode -wf** or equivalent SQL administration API command.

Usage

For security reasons, set the BAR_DEBUG_LOG configuration parameter to a directory with restricted permissions, such as the `$INFORMIXDIR` directory.

Related reference:

"BAR_DEBUG configuration parameter" on page 17-5

onmode -wf, -wm: Dynamically change certain configuration parameters (Administrator's Reference)

"PSM_DEBUG_LOG configuration parameter" on page 17-33

BAR_HISTORY configuration parameter

Use the BAR_HISTORY configuration parameter to specify whether the **sysutils** database maintains a backup history when you use **onsmsync** to expire old backups.

onconfig.std value
: Not in the `onconfig.std` file.

default value
: 0

range of values
: 0 = Remove records for expired backup objects from the **sysutils** database

  1 = Keep records for expired backup objects in the **sysutils** database

takes effect
: When **onsmsync** starts

  When you reset the value dynamically in your `onconfig` file by running the **onmode -wf** or equivalent SQL administration API command.

Usage

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.

For more information about **onsmsync**, see "The onsmsync utility" on page 8-4.

Related reference:

onmode -wf, -wm: Dynamically change certain configuration parameters (Administrator's Reference)

BAR_IXBAR_PATH configuration parameter

Use the BAR_IXBAR_PATH configuration parameter to change the path and name of the ON-Bar boot file.

onconfig.std *value*

UNIX or Linux: `$INFORMIXDIR/etc/ixbar.servernum`

Windows: `%INFORMIXDIR%\etc\ixbar.servernum`

range of values

Full path name for the ON-Bar boot file

takes effect

When ON-Bar or **onsmsync** starts

When you reset the value dynamically in your `onconfig` file by running the **onmode -wf** or equivalent SQL administration API command.

Usage

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
```

Related reference:

onmode -wf, -wm: Dynamically change certain configuration parameters (Administrator's Reference)

BAR_MAX_BACKUP configuration parameter

Use the BAR_MAX_BACKUP parameter to specify the maximum number of parallel processes that are allowed for each ON-Bar command.

onconfig.std *value*

0

if value not present

4

units ON-Bar processes

values 0 = Maximum number of processes allowed on system

1 = Serial backup or restore

n = Specified number of processes created

takes effect

When ON-Bar starts

When you reset the value dynamically in your `onconfig` file by running the **onmode -wf** or equivalent SQL administration API command.

Although the database server default value for BAR_MAX_BACKUP is 4, the `onconfig.std` value is 0.

Both UNIX and Windows support parallel backups.

Specify 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.

Specify 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 creates 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.

Related reference:

onmode -wf, -wm: Dynamically change certain configuration parameters (Administrator's Reference)

BAR_NB_XPORT_COUNT configuration parameter

Use the BAR_NB_XPORT_COUNT configuration parameter to specify the number of data buffers that each **onbar_d** process can use to exchange data with the database server.

onconfig.std *value*

20

if value not present
20

units Buffers

range of values
3 to unlimited

takes effect
When ON-Bar starts

When you reset the value dynamically in your `onconfig` file by running the **onmode -wf** or equivalent SQL administration API command.

The value of this parameter affects ON-Bar performance. For example, if you set BAR_NB_XPORT_COUNT to 5 and then issue five ON-Bar commands, the resulting 25 ON-Bar processes use a total of 125 buffers.

To calculate the amount of memory that each **onbar_d** process requires, use the following formula. For information about the page size for your system, see the release notes:

```
required_memory = (BAR_NB_XPORT_COUNT * BAR_XFER_BUF_SIZE
                  * page_size) + 5 MB
```

Related reference:

onmode -wf, -wm: Dynamically change certain configuration parameters (Administrator's Reference)

BAR_PERFORMANCE configuration parameter

Use the BAR_PERFORMANCE configuration parameter to specify the type of performance statistics to report to the ON-Bar activity log for backup and restore operations.

onconfig.std *value*
0

units Levels of statistics

values 0 = Does not collect performance statistics

1 = Reports time spent transferring data between the Informix 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 ON-Bar starts

When you reset the value dynamically in your `onconfig` file by running the **onmode -wf** or equivalent SQL administration API command.

Usage

For example, if you set BAR_PERFORMANCE to 3, ON-Bar reports the time spent transferring data between the IBM Informix 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 Informix 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.

Related reference:

onmode -wf, -wm: Dynamically change certain configuration parameters (Administrator's Reference)

"View ON-Bar backup and restore performance statistics" on page 8-10

BAR_PROGRESS_FREQ configuration parameter

Use the BAR_PROGRESS_FREQ configuration parameter to specify, in minutes, the frequency of the progress messages in the ON-Bar activity log for backup and restore operations.

onconfig.std *value*
0

if value not present
0

units minutes

range of values
0, then 5 to unlimited

takes effect
When ON-Bar starts

When you reset the value dynamically in your `onconfig` file by running the **onmode -wf** or equivalent SQL administration API command.

Usage

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.

Related reference:

onmode -wf, -wm: Dynamically change certain configuration parameters (Administrator's Reference)

BAR_RETRY configuration parameter

Use the BAR_RETRY configuration parameter to specify how many times **onbar** should try a data backup, logical-log backup, or restore operation if the first attempt fails.

onconfig.std *value*
1

if value not present
1

units integer

range of values
0 = BAR_ABORT, stop the rest of the backup/restore

1 = BAR_CONT, continue the rest of the backup/restore

n = 2 to 32766

takes effect
When ON-Bar starts

When you reset the value dynamically in your `onconfig` file by running the **onmode -wf** or equivalent SQL administration API command.

Usage

The setting of the BAR_RETRY parameter determines ON-Bar behavior in the following ways:

- If set to 0 (BAR_ABORT), ON-Bar stops 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 1 (BAR_CONT), ON-Bar stops 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 (retry backup and restore operations 2 to 32766 times), 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.

Related reference:

onmode -wf, -wm: Dynamically change certain configuration parameters (Administrator's Reference)

"onbar -RESTART syntax: Restarting a failed restore" on page 6-18

"Resolve a failed restore" on page 6-20

BAR_SIZE_FACTOR configuration parameter

Use the BAR_SIZE_FACTOR configuration parameter to augment the estimate for the size of a backup object, before the backup.

onconfig.std *value*
0

range of values
Positive integer

takes effect
When the database server starts

When you reset the value dynamically in your `onconfig` file by running the **onmode -wf** or equivalent SQL administration API command.

Usage

The estimate is handled before 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 back up can change during the backup. Some storage managers are strict and if the backup estimate is too low, the backup results 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 you set this parameter 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.

Related reference:

onmode -wf, -wm: Dynamically change certain configuration parameters (Administrator's Reference)

BAR_XFER_BUF_SIZE configuration parameter

Use the BAR_XFER_BUF_SIZE configuration parameter to specify the size of each transfer buffer.

onconfig.std value

31 if the page size is 2 KB

15 if the page size is 4 KB

units pages

range of values

For storage managers that support long transfer buffers:

- 1 - 16383 pages when the page size is 4 KB
- 1 - 32766 pages when the page size is 2 KB

For storage managers that do not support long transfer buffers:

- 1 - 15 if the Informix base page size is 4 KB
- 1 - 31 if the Informix base page size is 2 KB

takes effect

When ON-Bar starts

When you reset the value dynamically in your `onconfig` file by running the **onmode -wf** or equivalent SQL administration API command.

Usage

The database server passes the transfer buffer to ON-Bar and the storage manager.

The value of BAR_XFER_BUF_SIZE is in Informix base page sizes. For Linux, Solaris, and HP, the Informix base page size is 2 KB; and for AIX, Windows, and Mac, the Informix base page size is 4 KB. To calculate the size of the transfer buffer in a storage space or logical-log backup, multiply the value of the BAR_XFER_BUF_SIZE configuration parameter by the system page size, as shown in the following formula:

```
one transfer buffer KB = BAR_XFER_BUF_SIZE * pagesize
```

You can determine the system page size by running the **onstat -b** command.

The maximum size of the transfer buffer for many storage managers is 64 KB. IBM Tivoli Storage Manager (TSM) and IBM Informix Primary Storage Manager support long transfer buffer sizes of up to 65532 KB.

To calculate how much memory, in KB, the database server needs for each transfer buffer, use the following formula:

```
memory KB = (BAR_XFER_BUF_SIZE * pagesize) + 500 bytes/1028
```

The extra 500 bytes is for system use. For example, if BAR_XFER_BUF_SIZE is 15, the transfer buffer can be 66,292 bytes, or 64.5 KB.

The number of transfer buffers per backup stream is specified by the value of the BAR_NB_XPORT_COUNT configuration parameter, and the number of parallel backup streams is specified by the BAR_MAX_BACKUP configuration parameter.

Restriction: You cannot change the buffer size between a backup and restore. The values of the AC_TAPEBLOCK and AC_LTAPEBLOCK configuration parameters must be the same value as the value of the BAR_XFER_BUF_SIZE configuration parameter was at the time of backup.

Example

For example, for a transfer buffer size of 128*2 KB (a value of 256 KB) on Linux, specify:

```
BAR_XFER_BUF_SIZE  128
```

Related concepts:

"Configuring ON-Bar for optional TSM features" on page 4-6

Related reference:

onmode -wf, -wm: Dynamically change certain configuration parameters (Administrator's Reference)

onstat -b command: Print buffer information for buffers in use (Administrator's Reference)

IFX_BAR_NO_BSA_PROVIDER environment variable

Set the **IFX_BAR_NO_BSA_PROVIDER** environment variable to force ON-Bar to use the `sm_versions` file as the source of information about the XBSA library for the storage manager.

By default, ON-Bar communicates directly with the XBSA library for some storage managers. ON-Bar does not require that the `sm_versions` file is updated to contain information about the XBSA library for those storage managers.

Set the **IFX_BAR_NO_BSA_PROVIDER** environment variable if you are instructed to do so by IBM Software Support.

To unset the **IFX_BAR_NO_BSA_PROVIDER** environment variable, run the following command:

```
unset IFX_BAR_NO_BSA_PROVIDER
```

IFX_BAR_NO_LONG_BUFFERS environment variable

Set the **IFX_BAR_NO_LONG_BUFFERS** environment variable to prevent the size of transfer buffers from exceeding 64 KB when the BAR_XFER_BUF_SIZE configuration parameter is set to a long transfer buffer size value.

IBM Tivoli Storage Manager (TSM) and IBM Informix Primary Storage Manager support long transfer buffer sizes of up to 65532 KB.

Set the **IFX_BAR_NO_LONG_BUFFERS** environment variable if you are instructed to do so by IBM Software Support.

To unset the **IFX_BAR_NO_LONG_BUFFERS** environment variable, run the following command:

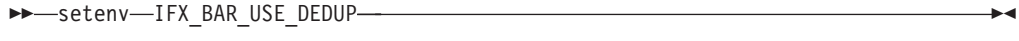
```
unset IFX_BAR_NO_LONG_BUFFERS
```

Related concepts:

"Configuring ON-Bar for optional TSM features" on page 4-6

IFX_BAR_USE_DEDUP environment variable

Set the **IFX_BAR_USE_DEDUP** environment variable to optimize the deduplication capabilities of storage managers.



You are not required to set the **IFX_BAR_USE_DEDUP** environment variable to a value. You can set it to any value or to no value.

Deduplication is a storage manager feature that reduces the size of backups by removing data that exists in older backups. To use deduplication, enable deduplication for your storage manager and set the **IFX_BAR_USE_DEDUP** environment variable in the Informix environment and then restart the database server. When you set the **IFX_BAR_USE_DEDUP** environment variable, the backup format has fewer unique pages, which optimizes the deduplication process.

Deduplication is incompatible with incremental backups. Do not make incremental backups while the **IFX_BAR_USE_DEDUP** environment variable is set.

Important: Backups that you take while the **IFX_BAR_USE_DEDUP** environment variable is set must be restored while the **IFX_BAR_USE_DEDUP** environment variable is set.

IBM Informix Primary Storage Manager does not support deduplication.

To unset the **IFX_BAR_USE_DEDUP** environment variable, run the following command:

```
unset IFX_BAR_USE_DEDUP
```

After you unset the **IFX_BAR_USE_DEDUP** environment variable, you must perform a level-0 backup.

Related concepts:

"Configuring ON-Bar for optional TSM features" on page 4-6

Related tasks:

"Configuring a third-party storage manager" on page 4-6

Related reference:

"Editing the TSM client system options file" on page 4-3

Chapter 11, "Configure ontape," on page 11-1

IFX_TSM_OBJINFO_OFF environment variable

Set the **IFX_TSM_OBJINFO_OFF** environment variable to disable support for restoring backup objects that are replicated, imported, or exported between IBM Tivoli Storage Manager (TSM) servers.

By default, ON-Bar stores unique IDs in the metadata of backup objects that are created by TSM. During a restore, ON-Bar checks the metadata to identify the object. Therefore, ON-Bar can restore a backup whose original ID, which is stored as CopyID columns in the ON-Bar catalog, changed because the object was replicated, exported, or imported between TSM servers. To prevent the ability to restore backup objects that are moved between TSM servers, set the **IFX_TSM_OBJINFO_OFF** environment variable to 1.

To unset the **IFX_TSM_OBJINFO_OFF** environment variable, run the following command:

```
unset IFX_TSM_OBJINFO_OFF
```

Related concepts:

"Configuring ON-Bar for optional TSM features" on page 4-6

LTAPEBLK configuration parameter

Use the LTAPEBLK configuration parameter to specify the block size of the device to which the logical logs are backed up when you use **ontape** for dbspace backups.

LTAPEBLK also specifies the block size for the device to which data is loaded or unloaded when you use the -l option of **onload** or **onunload**. If you are using **onload** or **onunload**, you can specify a different block size at the command line.

onconfig.std *value*

- On UNIX: 32
- On Windows: 16

units Kilobytes

range of values

Values greater than (*page size*/1024)

To obtain the page size, run the **onstat -b** command.

takes effect

For **ontape**:

- When you execute **ontape**.
- When you reset the value dynamically in your onconfig file by running the **onmode -wf** command.

For **onload** and **onunload**: When the database server is shut down and restarted

Usage

Specify LTAPEBLK as the largest block size permitted by your tape device. The database server does not check the tape device when you specify the block size. Verify that the LTAPEDEV tape device can read the block size that you specify. If not, you might not be able to read from the tape.

UNIX only: The UNIX **dd** utility can verify that the LTAPEDEV tape device can read the block size. It is available with most UNIX systems.

If you specify a LTAPEBLK value, ON-Bar ignores the value.

Related concepts:

The onunload and onload utilities (Migration Guide)

Related reference:

"Changing your ontape configuration" on page 11-6

onmode -wf, -wm: Dynamically change certain configuration parameters (Administrator's Reference)

"LTAPEDEV configuration parameter"

"LTAPESIZE configuration parameter" on page 17-18

Part 3, "ontape backup and restore system"

"TAPEBLK configuration parameter" on page 17-21

LTAPEDEV configuration parameter

Use the LTAPEDEV configuration parameter to specify the device or directory file system to which the logical logs are backed up when you use **ontape** for backups.

The LTAPEDEV configuration parameter also specifies the device to which data is loaded or unloaded when you use the **-l** option of **onload** or **onunload**. If you are using LTAPEDEV to specify a device for **onunload** or **onload**, the same information for TAPEDEV is relevant for LTAPEDEV.

onconfig.std *value*

On UNIX: **/dev/tapedev**On Windows: **NUL**

if not present

On UNIX: **/dev/null**On Windows:**NUL**

takes effect

For **ontape**:

- When you execute **ontape**, if set to a tape device.
- When the database server is shut down and restarted, if set to **/dev/null** on UNIX or **nul** on Windows.
- When you reset the value dynamically in your `onconfig` file by running the **onmode -wf** command.
- When you reset the value in memory by running the **onmode -wm** command.

For **onload** and **onunload**: When the database server is shut down and restarted

Usage

Warning: Do not set LTAPEDEV to **/dev/null** or **nul** when you use ON-Bar to back up logical logs.

If you specify a tape device in the LTAPEDEV configuration parameter, ON-Bar ignores the value.

Important: Set LTAPEDEV to `/dev/null` or leave it blank on UNIX or `NUL` on Windows only if you do not want to back up the logical logs. You must take the database server offline before you change the value of LTAPEDEV to `/dev/null`.

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.
- The ON-Bar activity log shows 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 are not 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. .

Related concepts:

The onunload and onload utilities (Migration Guide)

Related reference:

"Changing your ontape configuration" on page 11-6

"LTAPEBLK configuration parameter" on page 17-16

onmode -wf, -wm: Dynamically change certain configuration parameters (Administrator's Reference)

Part 3, "ontape backup and restore system"

"TAPEDEV configuration parameter" on page 17-21

LTAPESIZE configuration parameter

Use the LTAPESIZE configuration parameter to specify the maximum tape size of the device to which the logical logs are backed up when you use **ontape** for backups.

The LTAPESIZE configuration parameter also specifies the maximum tape size of the device to which data is loaded or unloaded when you use the **-l** option of **onload** or **onunload**. If you are using **onload** or **onunload**, you can specify a different tape size on the command line. If you want to use the full capacity of a tape, set LTAPESIZE to 0.

onconfig.std *value*
`0`

units Kilobytes

range of values
`0` or any positive number. The real value operating system dependent.

takes effect

For **ontape**:

- When you execute **ontape**.
- When you reset the value dynamically in your `onconfig` file by running the **onmode -wf** command.

For **onload** and **onunload**: When the database server is shut down and restarted

Usage

LTAPESIZE specifies the maximum tape size of the device to which the logical logs are backed up when you use **ontape** for backups. LTAPESIZE also specifies the maximum tape size of the device to which data is loaded or unloaded when you use the **-l** option of **onload** or **onunload**. If you are using **onload** or **onunload**, you can specify a different tape size on the command line. If you want to use the full capacity of a tape, set LTAPESIZE to 0.

Note: If the BACKUP_FILTER parameter is set in the ONCONFIG file, the LTAPESIZE cannot be set to 0. Otherwise the **ontape** utility returns an error when backing up logical logs to a directory on disk. The error message is:

```
The LTAPESIZE configuration parameter cannot be set to 0 when the BACKUP_FILTER
configuration parameter is set; change the value of LTAPESIZE.
Program over.
```

A workaround is to set the LTAPESIZE configuration parameter to a very high value. Log files are not much higher than the LOGSIZE configuration parameter. Use the value in the LOGSIZE as the upper limit for this database.

If you specify a LTAPESIZE value, ON-Bar ignores the value.

Related concepts:

The onunload and onload utilities (Migration Guide)

Related reference:

"Changing your ontape configuration" on page 11-6

"LTAPEBLK configuration parameter" on page 17-16

onmode -wf, -wm: Dynamically change certain configuration parameters (Administrator's Reference)

Part 3, "ontape backup and restore system"

"TAPESIZE configuration parameter" on page 17-23

RESTARTABLE_RESTORE configuration parameter

Use the RESTARTABLE_RESTORE configuration parameter to enable or disable restartable restores.

onconfig.std value

RESTARTABLE_RESTORE ON

values

OFF Disables restartable restore. If a restore fails and RESTARTABLE_RESTORE is OFF, you are not 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

After you edit your `onconfig` file. 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 "onbar -RESTART syntax: Restarting a failed restore" on page 6-18.

Related reference:

"onbar -RESTART syntax: Restarting a failed restore" on page 6-18

"Resolve a failed restore" on page 6-20

RESTORE_FILTER configuration parameter

Use the RESTORE_FILTER configuration parameter to specify the path name of a filter program, and any options.

onconfig.std value

Not set. Restored data is not filtered.

values The path name of a command and any options. By default, the path name is relative to the `$INFORMIXDIR/bin` directory, otherwise, the path name must be the absolute path of the program. If you include command-line options, both the filter name and the options must be surrounded by single quotation marks.

takes effect

After you edit your `onconfig` file and ON-Bar or **ontape** starts.

Usage

This filter transforms data that was transformed during backup to its original format prior to a restore. The filter specified by the RESTORE_FILTER configuration parameter must match the filter specified by the BACKUP_FILTER configuration parameter. For example, if data was compressed during backup, data must be uncompressed during a restore.

For security purposes, filters should not have write permission to non-privileged users. Permission on the filters are the same as that of permission on other executable files that are called by the IBM Informix server or utilities.

For example, if you want to compress backed up data, you could set the BACKUP_FILTER and RESTORE_FILTER configuration parameters to the following values:

```
BACKUP_FILTER /bin/compress
RESTORE_FILTER /bin/uncompress
```

The RESTORE_FILTER configuration parameter can include command-line options as well as the filter name. 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.

Related reference:

"BACKUP_FILTER configuration parameter" on page 17-2

TAPEBLK configuration parameter

Use the TAPEBLK configuration parameter to specify the block size of the device to which **ontape** writes during a storage-space backup.

onconfig.std *value*

- On UNIX: 32
- On Windows: 16

units Kilobytes

range of values
Values greater than `pagesize/1024`

To obtain the page size, run the **onstat -b** command.

takes effect
For **ontape**:

- When you execute **ontape**.
- When you reset the value dynamically in your `onconfig` file by running the **onmode -wf** command.
- When you reset the value in memory by running the **onmode -wm** command.

For **onload** and **onunload**: When the database server is shut down and restarted

Usage

TAPEBLK also specifies the default block size of the device to which data is loaded or unloaded when you use the **onload** or **onunload** utilities. If you are using **onload** or **onunload**, you can specify a different block size on the command line.

The database server does not check the tape device when you specify the block size. Verify that the TAPEBLK tape device can read the block size that you specify. If not, you might not able to read from the tape.

If you specify a TAPEBLK value, ON-Bar ignores the value.

Related concepts:

The onunload and onload utilities (Migration Guide)

Related reference:

"Changing your ontape configuration" on page 11-6

onmode -wf, -wm: Dynamically change certain configuration parameters (Administrator's Reference)

"TAPEDEV configuration parameter"

"TAPESIZE configuration parameter" on page 17-23

"LTAPEBLK configuration parameter" on page 17-16

Part 3, "ontape backup and restore system"

TAPEDEV configuration parameter

Use the TAPEDEV configuration parameter to specify the device or directory file system to which the **ontape** utility backs up storage spaces.

onconfig.std *value*
On UNIX: **/dev/tapedev**

On Windows: **\\.\TAPE0**

if not present
On UNIX: **/dev/null**

On Windows: **NUL**

units Path name

takes effect
For the **ontape** utility:

- If it is set to `/dev/null` on UNIX or `NUL` on Windows, when the database server is shut down and restarted
- If it is set to a tape device, when you run the **ontape** utility
- When you reset the value dynamically in your `onconfig` file by running the **onmode -wf** command.
- When you reset the value in memory by running the **onmode -wm** command.

For the **onload** and **onunload** utilities: When the database server is shut down and restarted

Usage

The **ontape** utility reads the value of the TAPEDEV parameter at the start of processing. If you set TAPEDEV to `/dev/null`, you must do it before you start **ontape** to request the backup. 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.

You can set the TAPEDEV configuration parameter to `STDIO` to direct **ontape** utility back up and restore operations to standard I/O instead of to a device.

The TAPEDEV configuration parameter also specifies the default device to which data is loaded or unloaded when you use the **onload** or **onunload** utilities. However, if TAPEDEV is set to `STDIO`, the **onunload** utility will not be able to unload data.

If you change the tape device, verify that the TAPEBLK and TAPESIZE configuration parameter values are correct for the new device.

If you specify a TAPEDEV value, ON-Bar ignores the value.

Remote devices (UNIX)

You can perform a storage-space backup across your network to a remote device attached to another host computer on UNIX and Linux platforms. The remote device and the database server computer must have a trusted relationship so that the **rsh** or the **rlogin** utility can connect from the database server computer to the remote device computer without asking for password. You can establish a trusted relationship by configuring the `/etc/hosts.equiv` file, the user's `~/.rhosts` files, or any equivalent mechanism for your system on the remote device computer. If you want to use a different utility to handle the remote session than the default utility used by your platform, you can set the **DBREMOTECMD** environment variable to the specific utility that you want to use.

Symbolic links to remote devices (UNIX)

The TAPEDEV configuration parameter can be a symbolic link, enabling you to switch between tape devices without changing the path name that the TAPEDEV configuration parameter specifies.

Use the following syntax to specify a tape device attached to another host computer:

host_machine_name:tape_device_pathname

The following example specifies a tape device on the host computer **kyoto**:

```
kyoto:/dev/rmt01
```

Rewinding tape devices before opening and on closing

The tape device that The TAPEDEV configuration parameter specifies must perform a rewind before it opens and when it closes. The database server requires this action because of a series of checks that it performs before it writes to a tape.

When the database server attempts to write to any tape other than the first tape in a multivolume dbspace or logical-log backup, the database server first reads the tape header to make sure that the tape is available for use. Then the device is closed and reopened. The database server assumes the tape was rewound when it closed, and the database server begins to write.

Whenever the database server attempts to read a tape, it first reads the header and looks for the correct information. The database server does not find the correct header information at the start of the tape if the tape device did not rewind when it closed during the write process.

Related concepts:

The onunload and onload utilities (Migration Guide)

Related reference:

"Changing your ontape configuration" on page 11-6

"TAPEBLK configuration parameter" on page 17-21

onmode -wf, -wm: Dynamically change certain configuration parameters (Administrator's Reference)

"LTAPEDEV configuration parameter" on page 17-17

Part 3, "ontape backup and restore system"

TAPESIZE configuration parameter

Use the TAPESIZE parameter specifies the size of the device to which **ontape** backs up storage spaces.

onconfig.std *value*
: `0`

units
: Kilobytes

range of values
: `0` or any positive number. The real value operating system dependent.

takes effect
: For **ontape**:
  - When you execute **ontape**.

- When you reset the value dynamically in your onconfig file by running the **onmode -wf** command.
- When you reset the value in memory by running the **onmode -wm** command.

For **onload** and **onunload**: When the database server is shut down and restarted

Usage

The TAPESIZE also specifies the size of the default device to which data is loaded or unloaded when you use **onload** or **onunload**. If you are using **onload** or **onunload**, you can specify a different tape size on the command line. If you want to use the full physical capacity of a tape, set TAPESIZE to 0.

Note: Tape size is irrelevant if TAPEDEV is set to STDIO.

If you specify a TAPESIZE value, ON-Bar ignores the value.

Related concepts:

The onunload and onload utilities (Migration Guide)

Related reference:

"Changing your ontape configuration" on page 11-6

"TAPEBLK configuration parameter" on page 17-21

onmode -wf, -wm: Dynamically change certain configuration parameters (Administrator's Reference)

Part 3, "ontape backup and restore system"

"LTAPESIZE configuration parameter" on page 17-18

The archecker utility configuration parameters and environment variable

These topics describe the **AC_CONFIG** environment variable and the configuration parameters that you use with the **archecker** utility.

The **archecker** utility uses the configuration parameters in the ac_config.std template to verify a backup or perform a table-level restore. If you need to change these parameters, copy the ac_config.std template to the AC_CONFIG file. The AC_CONFIG environment variable specifies the location of the AC_CONFIG file.

Because ON-Bar calls the **archecker** utility to verify backups, you must configure the **archecker** environment variable and parameters before you can use the **onbar -v** option.

You can also use other **archecker** configuration parameters that do not have default have default values in the ac_config.std file, but are valid in that file.

Table 17-1. Configuration parameters that the **archecker** *utility uses*

Configuration parameter	Description
AC_DEBUG	Prints debugging messages in the **archecker** message log.

Table 17-1. Configuration parameters that the **archecker** *utility uses (continued)*

Configuration parameter	Description
AC_IXBAR	Specifies the path name to the IXBAR file. If not set in the ac_config file, the value of the BAR_IXBAR_PATH configuration parameter is used.
AC_LTAPEBLOCK	Specifies the **ontape** block size for reading logical logs. If not set in the ac_config file, the value of the LTAPEBLOCK configuration parameter is used.
AC_LTAPEDEV	Specifies the local device name used by **ontape** for reading logical logs. If not set in the ac_config file, the value of the LTAPEDEV configuration parameter is used.
AC_MSGPATH	Specifies the location of the **archecker** message log. This configuration parameter is in the default ac_config file.
AC_SCHEMA	Specifies the path name to the **archecker** schema command file.
AC_STORAGE	Specifies the location of the temporary files that **archecker** builds. This configuration parameter is in the default ac_config file.
AC_TAPEBLOCK	Specifies the tape block size in kilobytes. If not set in the ac_config file, the value of the TAPEBLOCK configuration parameter is used.
AC_TAPEDEV	Specifies the local device name used by the **ontape** utility. If not set in the ac_config file, the value of the TAPEDEV configuration parameter is used.
AC_TIMEOUT	Specifies the timeout value for the **onbar** and the **archecker** processes if one of them exits prematurely.
AC_VERBOSE	Specifies either verbose or terse mode for **archecker** messages. This configuration parameter is in the default ac_config file.
BAR_BSALIB_PATH	Identical to the BAR_BSALIB_PATH server configuration parameter that is in the onconfig.std file. For more information, see "BAR_BSALIB_PATH configuration parameter" on page 17-4.

If you use the **ontape** utility and the AC_TAPEDEV, AC_TAPEBLK, AC_LTAPEDEV and AC_LTAPEBLK configuration parameters are not set in the AC_CONFIG file, the **archecker** utility will use the values specified in the TAPEDEV, TAPEBLK, LTAPEDEV, LTAPEBLK configuration parameters specified in the onconfig file.

Related reference:

Chapter 16, "archecker table level restore utility," on page 16-1

AC_CONFIG file environment variable

Set the **AC_CONFIG** environment variable to the full path name for the **archecker** configuration file (either ac_config.std or user defined).

default *value*
UNIX: `$INFORMIXDIR/etc/ac_config.std`

Windows: `%INFORMIXDIR%\etc\ac_config.std`

takes effect
When ON-Bar starts

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.

Important: If you do not specify the entire path, including the configuration file name in the AC_CONFIG file, the **archecker** utility might not work correctly.

AC_DEBUG configuration parameter

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 can substantially slow down **archecker** processing.

Default value
Off

Range 1-16

AC_IXBAR configuration parameter

Use the AC_IXBAR configuration parameter to specify the location of the `IXBAR` file.

Default value
None

Range Any valid path name

AC_LTAPEBLOCK configuration parameter

Use the AC_LTAPEBLOCK configuration parameter to the **ontape** block size for reading logical logs.

Default value
32 kilobytes

Range 0 - 2,000,000,000

Usage

When you perform an archive with:

- **onbar -b**, the value of AC_TAPEBLOCK should be the value the BAR_XFER_BUF_SIZE configuration parameter multiplied by the current page size. For more information, see "BAR_XFER_BUF_SIZE configuration parameter" on page 17-13.
- **ontape -t**, 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 "Specify the tape-block-size" on page 11-5.

AC_LTAPEDEV parameter

Use the AC_LTAPEDEV configuration parameter to specify the local device name that is used by the **ontape** utility.

If the tape device is set to STDIO, **archecker** receives input from standard input.

Default value
None

Range Any valid path name or STDIO

AC_MSGPATH configuration parameter

Use the AC_MSGPATH parameter in the **AC_CONFIG** file to specify the location of the **archecker** message log (`ac_msg.log`).

ac_config.std value
UNIX: `AC_MSGPATH /tmp/ac_msg.log`

Windows: `AC_MSGPATH c:\temp\ac_msg.log`

takes effect
When ON-Bar starts

Usage

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 try another backup, or give the `ac_msg.log` to IBM Software Support. For sample messages, see "onbar -v syntax: Verifying backups" on page 5-12.

AC_SCHEMA configuration parameter

Use the AC_SCHEMA configuration parameter to specify the path name to the **archecker** schema command file.

Default value
None

Range Any valid path name

This configuration parameter is overridden by the `-f` *cmdfile* command line option.

AC_STORAGE configuration parameter

Use the AC_STORAGE configuration parameter in the **AC_CONFIG** file to specify the location of the directory where **archecker** stores its temporary files.

ac_config.std *value*
UNIX: `/tmp`

Windows: `c:\temp`

takes effect
When ON-Bar starts

Usage

You must specify the entire path of the storage location in the **AC_CONFIG** file or else the **archecker** utility might not work correctly.

The following table lists the directories and files that **archecker** builds. If verification is successful, these files are deleted.

Table 17-2. The 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

To calculate the amount of free space that you need, see "Temporary space for backup verification" on page 5-14. It is recommended that you set AC_STORAGE to a location with plenty of free space.

AC_TAPEBLOCK configuration parameter

Use the AC_TAPEBLOCK configuration parameter to specify the size of the tape block in kilobytes when an archive is performed either the **onbar -b** command or the **ontape -t** command.

Default value
32 kilobytes

Range 0 - 2,000,000,000

Usage

When you perform an archive with:

- **onbar -b**, the value of AC_TAPEBLOCK should be the value the BAR_XFER_BUF_SIZE configuration parameter multiplied by the current page size. For more information, see "BAR_XFER_BUF_SIZE configuration parameter" on page 17-13.
- **ontape -t**, 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 "Specify the tape-block-size" on page 11-5.

AC_TAPEDEV configuration parameter

Use the AC_TAPEDEV configuration parameter to specify the local device name that is used by the **ontape** utility.

If the tape device is set to STDIO, **archecker** receives input from standard input.

Default value
None

Range Any valid path name or STDIO

AC_TIMEOUT configuration parameter

Use the AC_TIMEOUT configuration parameter to specify the timeout value for the **onbar** and the **archecker** processes if one of them exits prematurely.

ac_config.std *value*
UNIX: 300

Windows: 300

units seconds

takes effect
When the **onbar-v** command starts

The AC_TIMEOUT configuration 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 configuration parameter

Use the AC_VERBOSE parameter in the **AC_CONFIG** file to specify either verbose or terse output in the **archecker** message log (`ac_msg.log`).

ac_config.std *value*
1

range of values
1 = verbose messages in `ac_msg.log`

0 = terse messages in `ac_msg.log`

takes effect
When ON-Bar starts

Informix Primary Storage Manager configuration parameters

The IBM Informix Primary Storage Manager uses the information in some specific configuration parameters.

Related concepts:

Chapter 15, “Informix Primary Storage Manager,” on page 15-1

“Setting up Informix Primary Storage Manager” on page 15-8

Related tasks:

“Configuring Informix Primary Storage Manager” on page 15-9

“Examples: Manage storage devices with Informix Primary Storage Manager” on page 15-3

PSM_ACT_LOG configuration parameter

Use the PSM_ACT_LOG configuration parameter to specify the location of the IBM Informix Primary Storage Manager activity log if you do not want the log information included in the ON-Bar activity log.

onconfig.std *value*
none

if value not present
The value of the BAR_ACT_LOG configuration parameter is used

range of values
Full path name

takes effect
When the **onpsm** utility starts

When you reset the value dynamically in your `onconfig` file by running the **onmode -wf** command.

Usage

Specify a path to an existing directory with an appropriate amount of space available or use `$INFORMIXDIR/psm_act.log`. If you specify a file name only, the storage manager creates the activity log in the working directory in which you started the storage manager.

If the PSM_ACT_LOG configuration parameter is not set, the Informix Primary Storage Manager puts activity information in the directory specified with the BAR_ACT_LOG configuration parameter. To clearly distinguish ON-Bar and Informix Primary Storage Manager activity information, use the PSM_ACT_LOG to specify a different location for the storage manager 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 storage manager actions.

The file specified by the PSM_ACT_LOG configuration parameter is created if it does not exist.

You can also use the **PSM_ACT_LOG** environment variable to specify the location of the Informix Primary Storage Manager activity log for your environment, for example, for a single session.

Related reference:

"Message logs for Informix Primary Storage Manager" on page 15-19

"BAR_ACT_LOG configuration parameter" on page 17-3

PSM_ACT_LOG environment variable (SQL Reference)

onmode -wf, -wm: Dynamically change certain configuration parameters (Administrator's Reference)

PSM_CATALOG_PATH configuration parameter

Use the PSM_CATALOG_PATH configuration parameter to specify the full path to the directory that contains the IBM Informix Primary Storage Manager catalog tables. These catalog tables contain information about the pools, devices, and objects managed by the storage manager.

onconfig.std value
Not set. The default value is used.

default value
UNIX or Linux: `$INFORMIXDIR/etc/psm`

Windows: `%INFORMIXDIR%\etc\psm`

range of values
Full path name for the directory that contains the Informix Primary Storage Manager catalog tables

takes effect
When the ON-Bar or **onpsm** utility starts

When you reset the value dynamically in your `onconfig` file by running the **onmode -wf** command.

When you reset the value in memory by running the **onmode -wm** command.

Usage

You can change the default path to another location. The Informix Primary Storage Manager places the file that information about the devices and objects in whatever directory you specify.

If you move the backup file to another location, change the value of the PSM_CATALOG_PATH configuration parameter.

You can back up the contents of the file whenever you want.

If you have multiple instances, all instances contain the same catalog tables if the PSM_CATALOG_PATH in each instance is set to the same path. You can specify a different path for each instance.

The storage manager automatically creates the catalog tables the first time you run an **onpsm** utility command or the first time that the XBSA shared library is used.

You can also use the **PSM_CATALOG_PATH** environment variable to specify the location of the Informix Primary Storage Manager catalog tables for your environment, for example, for a single session.

Related reference:

PSM_CATALOG_PATH environment variable (SQL Reference)

onmode -wf, -wm: Dynamically change certain configuration parameters (Administrator's Reference)

PSM_DBS_POOL configuration parameter

Use the PSM_DBS_POOL configuration parameter to change the name of the pool in which the IBM Informix Primary Storage Manager places backup and restore dbspace data.

onconfig.std *value*
DBSPOOL

takes effect
When the **onpsm** utility starts

When you reset the value dynamically in your `onconfig` file by running the **onmode -wf** command.

When you reset the value in memory by running the **onmode -wm** command.

Usage

The storage manager automatically places the dbspace data in the DBSPOOL or the pool you specify. You can use any combination of letters and digits.

You can also use the **PSM_DBS_POOL**environment variable to change the name of the pool for your environment, for example, for a single session.

Related concepts:

"Device pools" on page 15-17

Related reference:

PSM_DBS_POOL environment variable (SQL Reference)

onmode -wf, -wm: Dynamically change certain configuration parameters (Administrator's Reference)

PSM_DEBUG configuration parameter

Use the PSM_DEBUG configuration parameter to specify the amount of debugging information that prints in the Informix Primary Storage Manager debug log if you want to use a debug level that is different from the one used by ON-Bar.

onconfig.std value
Not set. The default value is used.

default value
The value of the BAR_DEBUG configuration parameter is used.

units One digit to represent the level of debugging information that you want

range of values
0 = No debugging messages.

1 = Prints only internal errors.

2 = Prints information about the entry and exit of functions and prints internal errors.

3 = Prints the information specified by 1-2 with additional details.

4 = Prints information about parallel operations and the information specified by 1-3.

5 = Prints information about internal states in the Informix Primary Storage Manager.

6 = Prints the information specified by 1-5 with additional details.

7 = Prints information specified by 1-6 with additional details.

8 = Prints information specified by 1-7 with additional details.

9 = Prints all debugging information.

takes effect
When the **onpsm** utility starts

When the ON-Bar utility executes commands and reads information that is specified in the BAR_DEBUG configuration parameter

When you reset the value dynamically in your onconfig file by running the **onmode -wf** command.

When you reset the value in memory by running the **onmode -wm** command.

Usage

If you set the PSM_DEBUG configuration parameter to a valid value that is higher than `0`, the Informix Primary Storage Manager logs debug messages to its debug log.

You can experiment with the debug values to find the right amount of information. Generally, if the PSM_DEBUG configuration parameter is set to `5`, the storage manager prints enough information for tracing and debugging purposes.

Settings of `8` and `9` require a large amount of space.

You can also use the **PSM_DEBUG** environment variable to specify the amount of debugging information that prints in the storage manager debug log for your environment, for example, for a single session.

Related reference:

"Message logs for Informix Primary Storage Manager" on page 15-19

PSM_DEBUG environment variable (SQL Reference)

onmode -wf, -wm: Dynamically change certain configuration parameters (Administrator's Reference)

PSM_DEBUG_LOG configuration parameter

Use the PSM_DEBUG_LOG configuration parameter to specify the location of the debug log to which the IBM Informix Primary Storage Manager writes debugging messages if you do not want the log information included in the ON-Bar debug log.

onconfig.std value
: Not set. The default value is used.

default value
: The value of the BAR_DEBUG_LOG configuration parameter is used.

takes effect
: When the **onpsm** utility starts

  When you reset the value dynamically in your `onconfig` file by running the **onmode -wf** command.

  When you reset the value in memory by running the **onmode -wm** command.

Usage

If the PSM_DEBUG_LOG configuration parameter is not set, the Informix Primary Storage Manager puts activity information in the directory specified with the BAR_DEBUG_LOG configuration parameter. To clearly distinguish ON-Bar and Informix Primary Storage Manager activity information, use the PSM_DEBUG_LOG to specify a different location for the Informix Primary Storage Manager activity log.

For security reasons, set the PSM_DEBUG_LOG configuration parameter to a directory with restricted permissions, such as the `$INFORMIXDIR` directory.

If the directory that holds the debug file becomes too large, you can erase the file. You need to retain information only if there are problems that need to be debugged.

You can also use the **PSM_DEBUG_LOG** environment variable to specify the location of the debug log for your environment, for example, for a single session.

Related reference:

"Message logs for Informix Primary Storage Manager" on page 15-19

"BAR_DEBUG_LOG configuration parameter" on page 17-7

PSM_DEBUG_LOG environment variable (SQL Reference)

onmode -wf, -wm: Dynamically change certain configuration parameters (Administrator's Reference)

PSM_LOG_POOL configuration parameter

Use the PSM_LOG_POOL configuration parameter to change the name of the pool in which the IBM Informix Primary Storage Manager places backup and restore log data.

onconfig.std *value*
LOGPOOL

takes effect
When the **onpsm** utility starts

When you reset the value dynamically in your `onconfig` file by running the **onmode -wf** command.

When you reset the value in memory by running the **onmode -wm** command.

Usage

The storage manager automatically places the log data in the `LOGPOOL` or the pool you specify. You can use any combination of letters and digits.

You can also use the **PSM_LOG_POOL** environment variable to change the name of the pool for your environment, for example, for a single session.

Related concepts:

"Device pools" on page 15-17

Related reference:

PSM_LOG_POOL environment variable (SQL Reference)

onmode -wf, -wm: Dynamically change certain configuration parameters (Administrator's Reference)

Event alarm configuration parameters

When you set configuration parameters for use with the ON-Bar and **ontape** utilities, also determine if you need to adjust the ALARMPROGRAM and ALRM_ALL_EVENTS configuration parameters.

Use the ALARMPROGRAM configuration parameter to set the **log_full.sh** script to automatically back up log files when they become full.

Use the ALRM_ALL_EVENTS configuration parameter to cause ALARMPROGRAM to execute every time an alarm event is invoked.

Part 7. Appendixes

Appendix A. Troubleshooting some backup and restore errors

This appendix lists some error and informational messages that you can receive during a backup or restore, describes under what circumstances the errors might occur or the message might appear, and provides possible solutions or workarounds.

Find errors by viewing the ON-Bar activity log

The database server does not show errors in standard output (**stdout**) if an error occurs when you use **onbar -b** to back up 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`.

Corrupt page during an archive

The message `Archive detects that page is corrupt` indicates that page validation failed. If you receive this message, you can identify the table that has the corrupt page.

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 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 stops 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 informational messages `log backup is already running` in the `bar_act.log` file and `Process exited with return code 152` in the `online.log` file might appear under some circumstances.

These messages can appear under the following circumstances:

- When 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.
- When the **onbar -b -l** command is started automatically.

  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 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. If you receive these message, you can take steps to solve the problem.

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.

Drop 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. If you receive these message, you can take steps to solve the problem.

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.

Restore 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

In some circumstances when there is a problem with the system time, ON-Bar fails with the message `There are no storage spaces or logical logs to backup or restore`. If this occurs, you can take steps to solve the problem.

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. Migrate data, servers, and tools

Backing up before a database server or storage-manager upgrade

Before you upgrade to a new version of the database server, you must perform a complete backup.

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 or change storage managers.
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 complete information about database server migration, see the *IBM Informix Migration Guide*.

   If you change storage manager vendors, do not remove the old storage manager until you verify that the new storage manager works for both backup and restore operations.

Related reference:

"Upgrading a third-party storage manager"

"Changing storage-manager vendors" on page B-2

Upgrading a third-party storage manager

If you upgrade a third-party storage manager vendors, do not remove the old storage manager until you verify that the new storage manager works for both backup and restore operations.

Prerequisite: Before upgrading, perform a complete backup of the database server.

To use a new version of a third-party storage manager with the database server:

1. Install the new storage manager before you bring up the database server.
2. Update the `sm_versions` file with the new storage-manager definition.

Verify that the new storage manager operates correctly before you use it in a production environment:

- Make sure that the storage manager can find the backup objects that ON-Bar requests.
- Make sure that the new storage-manager version is able to read media written with your old version.

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.

Use the **onsmsync** utility to expire old backup history in the **sysutils** database and emergency boot files.

Related tasks:

"Backing up before a database server or storage-manager upgrade" on page B-1

Changing storage-manager vendors

If you change storage manager vendors, do not remove the old storage manager until you have proof that the new storage manager works for both backup and restore operations. You can use the old storage manager as a backup storage manager to use in case the new storage manager does not meet your needs.

ON-Bar supports working with multiple storage managers at the same time. To set up to test one storage manager and keep the other as a backup storage manager, specify information for both of the storage managers in the BAR_BSALIB_PATH configuration parameter and in the `$INFORMIXDIR/etc/sm_versions` file.

If you cannot use the old and new storage managers at the same time, use ON-Bar and the IBM Informix Primary Storage Manager or **ontape** as an alternative for backups while you check that backup and restore operations work correctly with the new storage manager. Until you confirm that the new storage manager works correctly, perform all your backups as whole system level-0 backups (**onbar -b -L 0 -w**)

If you change physical connectivity, such as moving a storage device from a local connection to a network server, make sure the new storage manager can move the data across the network. Also ensure that the new storage manager can send multiple data streams to storage devices. It also might use a different version of XBSA.

Related tasks:

"Backing up before a database server or storage-manager upgrade" on page B-1

Switching from ontape to ON-Bar

You cannot back up data with **ontape** and restore it with ON-Bar, or conversely because the data storage formats and backup capabilities are different. However, you can back up data with **ontape**, prepare to use ON-Bar, and then back up with ON-Bar.

To switch from **ontape** to ON-Bar:

1. Use **ontape** to perform a full backup.
2. Take the backup media offline to prevent possible reuse or erasure.
3. Configure the storage manager to be used with ON-Bar.
4. Configure your environment:
   a. Set the configuration parameters that you use with ON-Bar and the storage manager.
   b. If you use a storage manager other than the IBM Informix Primary Storage Manager, create the `sm_versions` file with the storage-manager definition. The Informix Primary Storage Manager does not use the **sm_versions.std** file.

5. Use ON-Bar (**onbar -b** or **onbar -b -w**) to perform a full backup.
6. Verify the backup with the **onbar -v** command.

Appendix C. GLS support

This appendix contains information about using Global Language Support (GLS) with ON-Bar.

Use GLS with the ON-Bar utility

The ON-Bar utility 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 globalization 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, `tomb`: **onbar -b -L 0 -f tomb**

On Windows, you cannot use multibyte file names in backup or restore commands because they are not supported.

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

You can use non-ASCII characters in the database names and filenames with the ON-Bar and **ondblog** commands, and for file names in the `onconfig` file.

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.

For example, you can specify a non-ASCII file name for the ON-Bar activity login 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 storage space names and 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 tells you that a database server error occurred appears in the client locale, and the server-generated part appears in the server locale.

Use 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.

Use GLS with the ontape utility

The **ontape** utility supports GLS in the same way as ON-Bar does. You can specify the database name in the national locale.

Appendix D. Accessibility

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

Accessibility features for IBM Informix products

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

Accessibility features

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

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

Keyboard navigation

This product uses standard Microsoft Windows navigation keys.

Related accessibility information

IBM is committed to making our documentation accessible to persons with disabilities. Our publications are available in HTML format so that they can be accessed with assistive technology such as screen reader software.

IBM and accessibility

For more information about the IBM commitment to accessibility, see the *IBM Accessibility Center* at http://www.ibm.com/able.

Dotted decimal syntax diagrams

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

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

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

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

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

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

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

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

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

! Specifies a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicates that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the same dotted decimal number can specify a ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In this example, if you include the FILE keyword but do not specify an option, default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP only applies to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

***** Specifies a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be

repeated. For example, if you hear the line `5.1* data-area`, you know that you can include more than one data area or you can include none. If you hear the lines `3*, 3 HOST`, and `3 STATE`, you know that you can include `HOST`, `STATE`, both together, or nothing.

Notes:

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

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

Notices

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

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy,

modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs.

© Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Privacy policy considerations

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM’s Privacy Policy at http://www.ibm.com/privacy and IBM’s Online Privacy Statement at http://www.ibm.com/privacy/details the section entitled “Cookies, Web Beacons and Other Technologies” and the “IBM Software Products and Software-as-a-Service Privacy Statement” at http://www.ibm.com/software/info/product-privacy.

Trademarks

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

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

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

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

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

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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

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

Index

Special characters

A

B

C

D

E

F

G

H

I

L

M

O

P

R

S

T

U

V

W

X

Printed in USA

SC27-4508-02

Spine information:

Informix Product Family Informix **Version 12.10**

IBM Informix Backup and Restore Guide