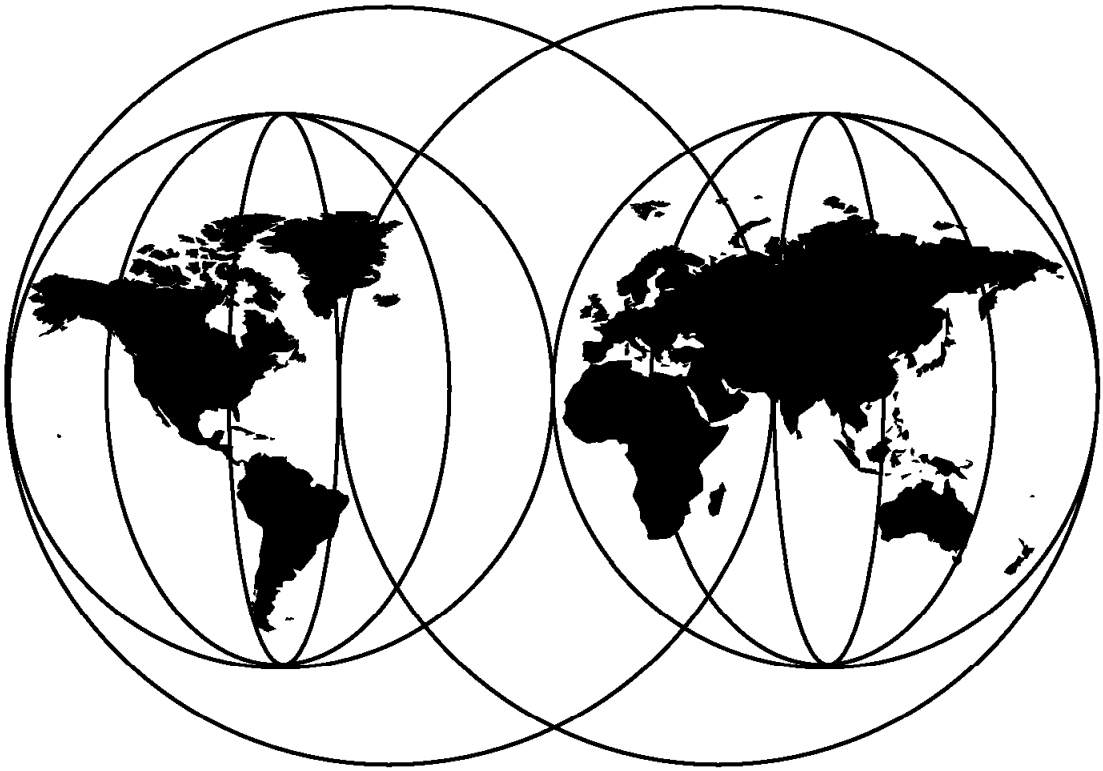
Using ADSM to Back Up Databases

Tim Mortimer, Dale McCurdy, Malkit Hayun, Frank Sarstedt, Georg Ember, Cyndie Behrens



International Technical Support Organization

http://www.redbooks.ibm.com

This book was printed at 240 dpi (dots per inch). The final production redbook with the RED cover will be printed at 1200 dpi and will provide superior graphics resolution. Please see "How to Get ITSO Redbooks" at the back of this book for ordering instructions.

SG24-4335-03

International Technical Support Organization

SG24-4335-03

Using ADSM to Back Up Databases

July 1998

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix F, “Special Notices” on page 573.

Fourth Edition (July 1998)

This edition applies to Version 3 of ADSM for AIX, Program Number 5765-C43; Version 3 of ADSM for MVS, Program Number 5655-A30; ADSM for HP-UX, Program Number 5639-D92; ADSM for SUN Solaris, Program Number 5639-D91. and ADSM for Windows NT, Program Number 5639-C59.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. QXXE Building 80-E2
650 Harry Road
San Jose, California 95120-6099

Contents

Preface

This book is written for IBM, customer, vendor, and consultant personnel who want to understand how to use ADSTAR Distributed Storage Manager (ADSM) to back up databases or learn how to integrate ADSM with database and operating system backup utilities. It is organized in four parts that describe techniques for backing up Oracle, Informix, Sybase, DB2/6000, and DB2/2 databases. The techniques are also applicable to other database products.

Part 1 is a primer on ADSM and databases. It provides an overview of ADSM, relational databases, and basic techniques for database backup and recovery.

Part 2 focuses on backup and recovery of Oracle7 and Oracle8 databases. It provides detailed examples of using the Oracle7 Enterprise Backup Utility (EBU) and Oracle8 Recovery Manager (RMAN) products with IBM′s ADSMConnect Agent, SQL-BackTrack from BMC Software, and IBM′s BACKINT/ADSM product for Oracle databases in a SAP R/3 environment.

Part 3 covers backup and recovery of Sybase and Informix databases using ADSM.

Part 4 covers backup and recovery of DB2/2 and DB2/6000 with ADSM. It also covers backup and recovery with IBM′s DataHub product.

ADSMPIPE, a tool used to pipe raw device data directly to ADSM, is described for use in backing up databases installed on UNIX raw logical partitions.

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization San Jose-Center.

Tim Mortimer is a Senior Systems Specialist at the International Technical Support Organization-San Jose Center. He writes extensively and teaches IBM classes worldwide on all areas of ADSM and distributed storage management. Before joining the ITSO in 1996, Tim worked for IBM in the United Kingdom as a storage management consultant. He has 20 years of IT experience, including 10 years of storage management experience covering a wide range of mainframe and client/server platforms.

Dale McCurdy is a Services I/T Specialist with IBM Australia where he installs and supports RS/6000 and SP systems. He has 10 years of experience with AIX and the RS/6000 systems product line. During those 10 years Dale spent 3 years providing level 2 hardware support to the Asia Pacific south region. His areas of expertise include systems management and problem isolation and resolution.

Malkit Hayun is an Applications Engineer working for IBM Business Partner UDI in Israel. She has 3 years of experience in the storage solutions field, specializing in ADSM. Malkit holds a degree in economics from Tel Aviv University. Her areas of expertise include RS/6000 products and services including ADSM, AIX, and system support. Malkit teaches AIX and ADSM classes for IBM Israel. She is also responsible for sales and technical support for UDI.

Frank Sarstedt is a Storage Solution Consultant with IBM Germany. He holds a degree in Computer Science from the Berufsakademie in Stuttgart Germany. Frank has 5 years of experience with ADSM implementation, consultancy, and presales support. His areas of expertise include storage management, high-availability systems, and relational databases.

Georg Ember is an Advisory Systems Engineer supporting RS/6000 and SP systems in Germany. He holds a degree in computer science from the FH Nuernberg/Germany. Georg has 7 years of experience with AIX and RS/6000 systems and 4 years experience with ADSM for AIX. He has spent several years performing AIX and ADSM services and providing software and sales technical support to the Central Europe Region. His areas of expertise include systems and storage management and sales technical support. He contributed to the first two editions of this book.

Cyndie Behrens is a Storage Management Consultant at IBM San Jose′s Executive Customer Briefing Center where she regularly presents to customers on all aspects of enterprise data management. She previously worked as a Distributed Storage Software Specialist at IBM′s ITSO, San Jose-Center, where she wrote and taught IBM classes worldwide on all areas of ADSM. Cyndie has written extensively on several aspects of ADSM but is most noted for her expertise in ADSM backup of databases (such as Oracle, Sybase, Informix, DB2, Lotus Notes, MS SQL Server, MS Exchange, and SAP R/3). Before moving to San Jose in 1993, Cyndie worked in software marketing and technical support in Los Angeles, California as a client/server specialist. She holds degrees in mathematics and psychology from the State University of New York at Binghamton.

The authors of the previous editions of this redbook were:

Carmen Banegas
IBM Venezuela

Leigh Beal
Mainland Information Services, Canada

Sanjay Bhat
IBM Texas

Susan Liston
IBM Maryland

Tim Mather
Mainland Information Services, Canada

Andy Wharton
IBM UK

Norman Taylor
IBM UK

Georg Ember
IBM Germany

Alessandro Rigobello
IBM Italy

Lazaro Flores
Infomedia, Mexico City

Thanks to the following people for their invaluable contributions to this project:

Thiha Than
IBM San Jose

Randy Helmonds
BMC Software, Inc.

Maggie Cutler
International Technical Support Organization, San Jose Center

Comments Welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in “ITSO Redbook Evaluation” on page 587 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Web sites:

  For Internet users `http://www.redbooks.ibm.com/`
  For IBM Intranet users `http://w3.itso.ibm.com/`

- Send us a note at the following address:

  `redbook@us.ibm.com`

Part 1. ADSM and Database Primer

Chapter 1. Introduction to ADSM

This chapter provides a brief overview of ADSM, the market′s best storage manager in a distributed environment. We look at the many platforms ADSM supports, the main components of ADSM, and descriptions of key functions such as scheduling and policy management. We conclude with a summary of ADSM advantages.

ADSTAR

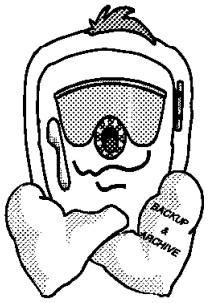
Distributed Storage Manager

The market's best

Storage Manager

in a

Distributed Environment



1.1 What Is ADSM?

ADSM, IBM′s solution to enterprisewide distributed storage management, is a client/server program product. It provides highly automated, centrally scheduled, network-based backup and archive functions for workstations and LAN file servers. ADSM supports a wide variety of IBM and non-IBM clients and servers, as shown in Figure 1 on page 4, and addresses the need for customer asset protection and data availability for distributed environments.

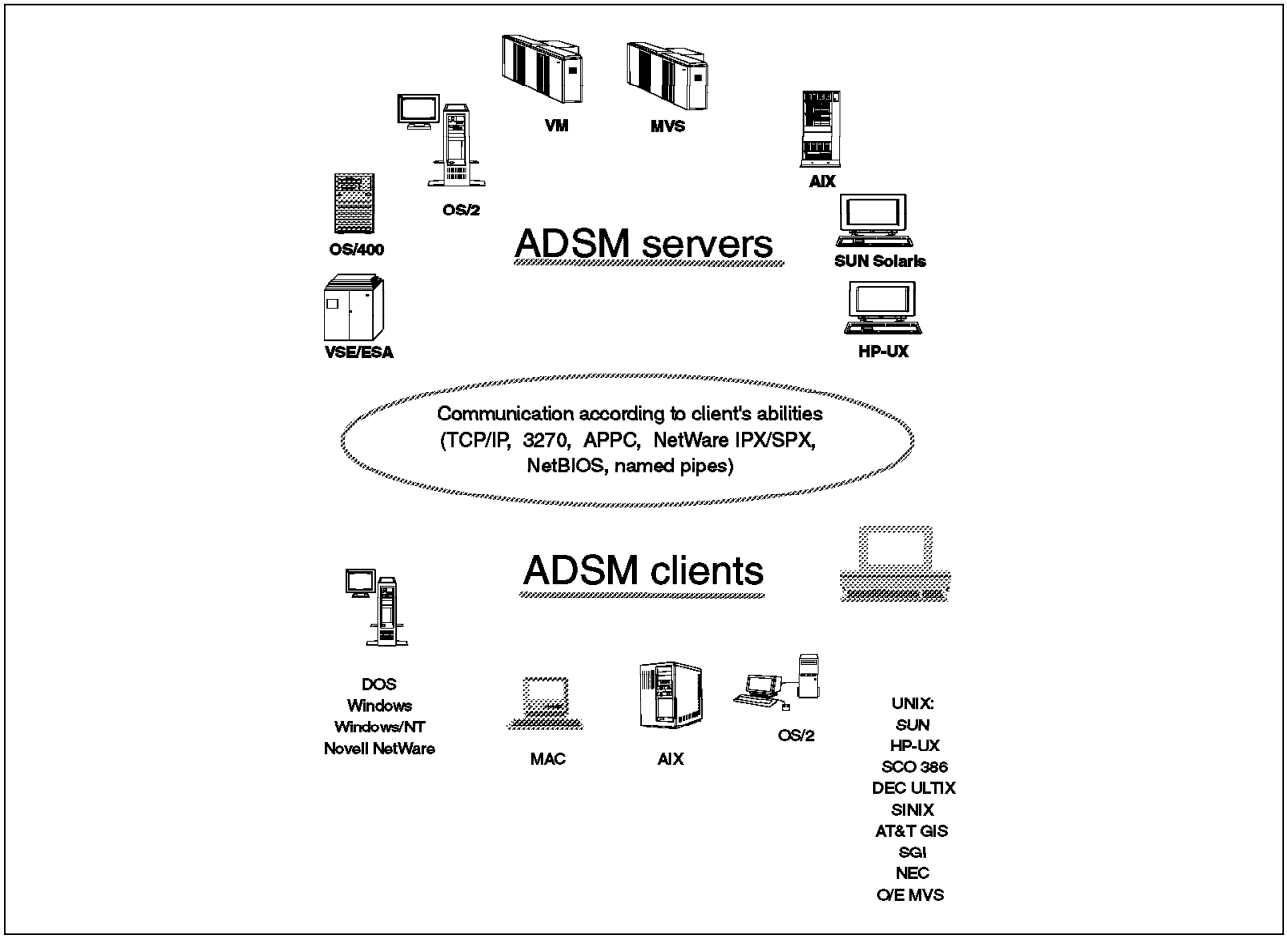


Figure 1. ADSM Platform Support

1.2 Main Components

Let us look at the main components of ADSM—the backup/archive client, the administrative client, and the server, as shown in Figure 2 on page 5—and then briefly review the application client.

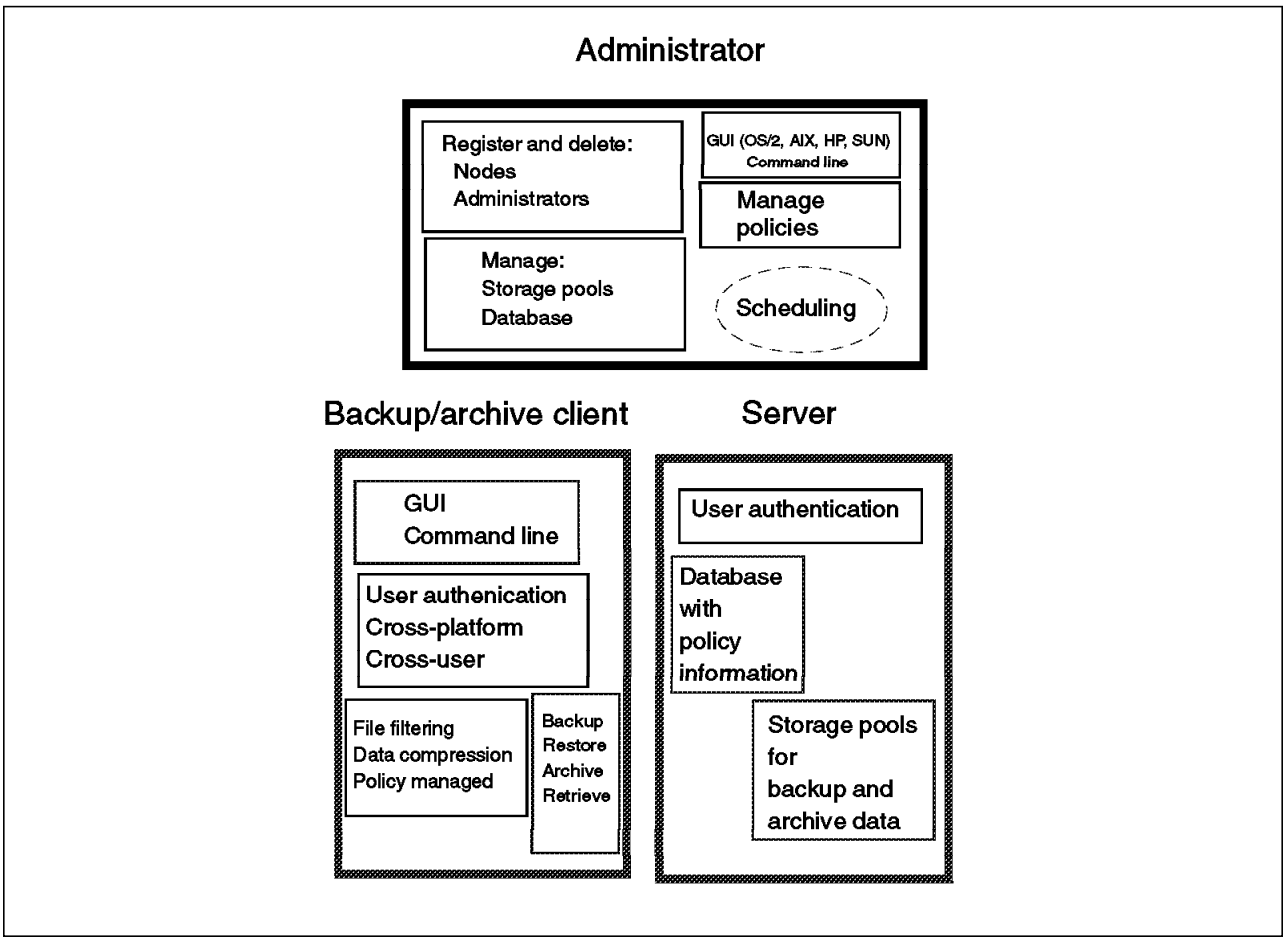


Figure 2. ADSM Storage Management Components

1.2.1 Backup/Archive Client

The backup/archive client runs on the workstation and, depending on the platform, provides both a graphical user interface (GUI) and command line interface (CLI). Although all clients are similar, they each have the look and feel of the platform on which they are running. Thus users can back up or restore files using an interface with which they are familiar.

The main function of ADSM is backup and restore. You can back up all of your files (full), specific files you want to back up (selective), or only those files that have changed since your last backup (incremental).

The file compression provided on the client platforms reduces network traffic and the amount of storage required on the server to store the files.

You can specifically include or exclude certain files from being backed up. For example, you might not want everyone to back up their local copies of the OS/2 operating system!

ADSM′s cross-user and cross-platform restore provide you with significant flexibility. Cross-user restore enables you to authorize someone else to restore your files. Cross-platform restore enables you to restore your file on a platform different from the platform on which it was backed up. For example, you could back up your file from a DOS workstation but then restore it to an OS/2

workstation. Cross-platform restore can be extremely useful when you migrate to new workstation platforms, or even if you happen to work at a different office one day that has different workstations. You will still have access to the data you backed up!

A separate archive/retrieve function is also part of ADSM. This function provides a way for you to store files that you may not use but need to retain for long-term storage. Archive is also useful as a way of reducing the disk space on your workstation. You can archive files for long-term storage and erase the original files from your workstation to create room for more active files and applications.

1.2.2 Administrative Client

As shown in Figure 3, an administrator controls or monitors server activity, defines storage management policies for workstation files, and sets up schedules to provide backup and archive services at regular intervals.

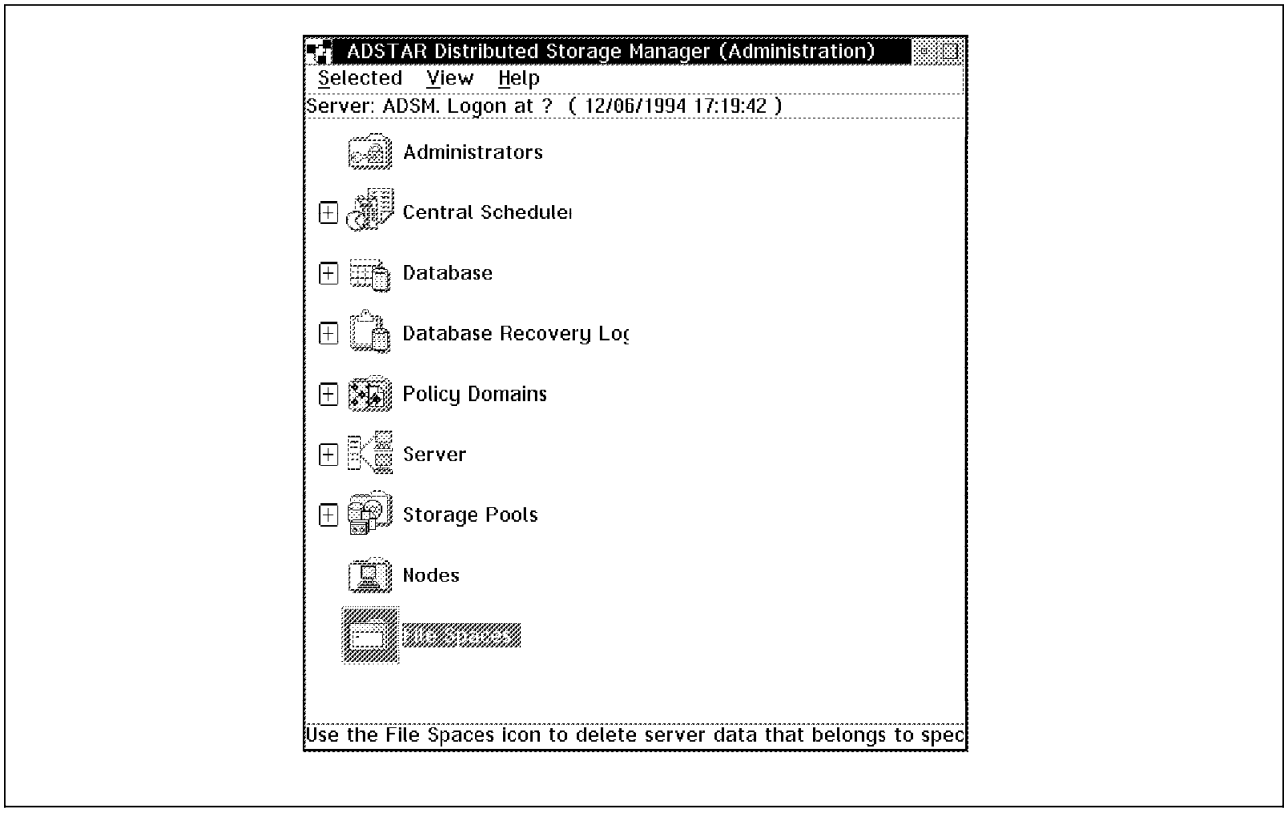


Figure 3. ADSM Administrative Client

An administrative client is a program that allows administrators to control and monitor the server through administrative commands. The administrative program can be installed on a programmable workstation (PWS), personal computer, or mainframe. An administrative client passes commands through an administrative command line. In some cases, a GUI has been added to the administrative client code.

ADSM provides a hierarchical structure to the authority you can grant an administrator. Thus you can establish as flexible an administration scheme as you would like while still providing control over your system. The ADSM administrator with overall authority is called the system administrator. The other administrators are called policy, storage, operator, or analyst administrators,

depending on which part of the system they control. Their administrative tasks are separated into logical categories, such as controlling the management policies, the storage pools and databases, the operation of the server, and the analysis of certain server events.

Dividing up the administrative authority based on logical categories of tasks is not the only way of granting authority. You can also divide up the administrative authority by organization. You can give the logical categories of authority to a department, but only for the data that belongs to that department. For example, you can give a department policy and storage authority for the policy domain and storage pools that it owns.

1.2.3 Server

The server component provides storage resources and services for the backup/archive clients. Users can back up or archive their files onto server storage resources, such as disk, tape, or optical devices that are managed and monitored by ADSM server policy.

Figure 4 shows the two key components of the ADSM server: the storage pools where the client files are actually stored, and the database that serves as an inventory or index to the client files within the storage pools. The database consists of the database space and the recovery log. The recovery log keeps track of all changes made to the database.

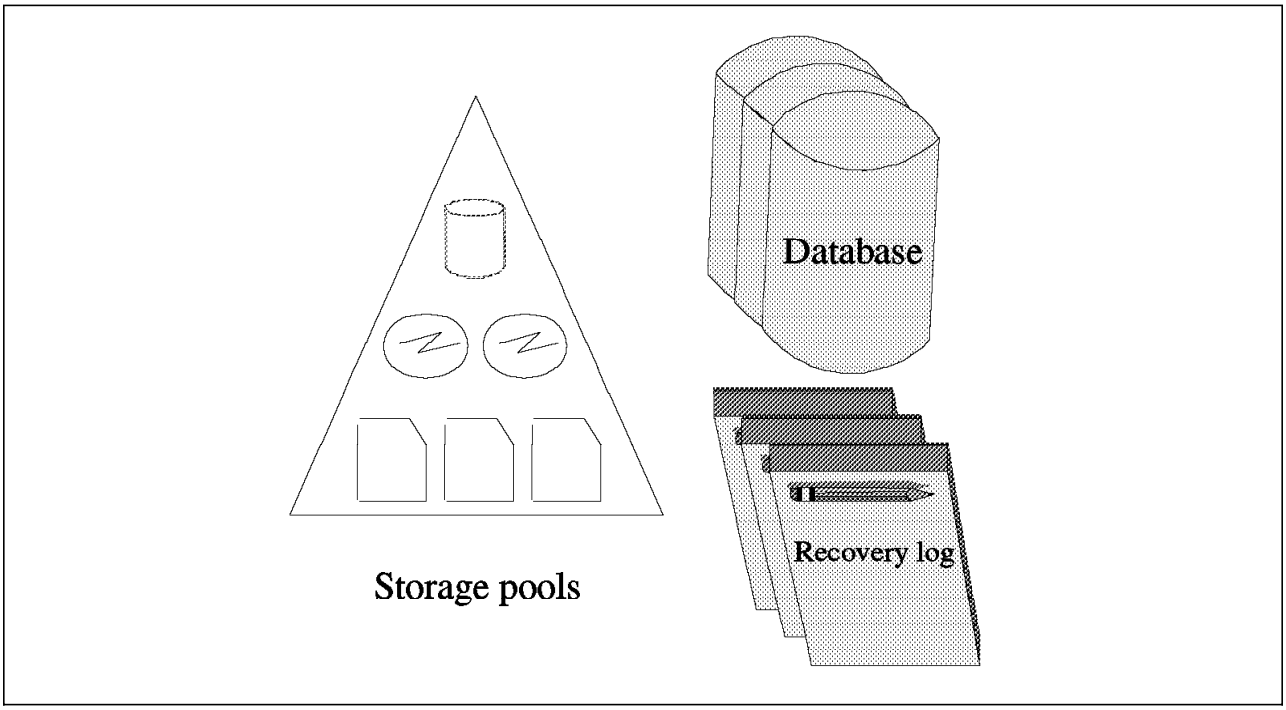


Figure 4. ADSM Server Components

The storage pools contain the client files that have been backed up or archived. A hierarchy of storage media can be used to define the storage pools. The pools can contain disk storage, optical devices, and tape devices. Each ADSM server platform supports a different set of storage media, so please verify the devices that are supported in your environment.

Data can be moved automatically through the storage hierarchy onto less expensive media with ADSM′s migration function. Additional management functions are provided, such as reclamation and collocation for tape management.

The ADSM server is multitasking, so multiple clients can back up data concurrently.

The ADSM database is the heart of the server. The server database is critical to the operation of ADSM because it contains file location information as well as policy and scheduling information. The following information is stored in the database:

- Information about registered client nodes
- Policies assigned to those client nodes
- Schedules and their association with client nodes
- Event records, such as whether a schedule successfully completed
- The activity log that contains the messages generated by the server
- Information about ADSM volumes
- The data storage inventory, that is, the information used to locate files that reside in storage pools.

The database has all of the features associated with a database management system. Because the database is critical, many features are built in to ADSM to help maintain the availability, integrity, and performance of the database. Two of these features are the recovery log and mirroring.

A recovery log is used to help maintain the integrity of the database. It keeps track of all changes made to the database, so that if a system outage were to occur, a record of the changes would be available in the log. When a change to the database occurs, the recovery log is updated with some transaction information before the database is updated. Thus uncommitted transactions can be rolled back during recovery so that the database remains consistent.

The administrator can configure the server so that up to three copies of the database and recovery logs are maintained at all times. This *mirroring* capability provides nondisruptive and immediate recovery from physical failures on database and recovery log volumes. Mirroring is the process of writing the same data to multiple storage devices at the same time.

If a mirrored volume encounters a media failure, the server automatically places the failing volume offline and continues database operations using the other mirrored copies. Once the failed disk is replaced and made available to the server, it is automatically synchronized with the intact copies.

The mirroring facility improves database performance. The mirrored copies are treated equally; there is no concept of primary copy and alternate copies. Therefore, the server reads from the database copy that is on the device with the best response time.

Another server function, export/import, creates a self-describing copy of specified server information. Information that can be exported includes:

- Administrator information

- Client node definitions
- Policy information
- Backup and archive data.

Export/import is useful for migration and conversion, workload balancing, as well as cloning of information.

ADSM provides extensive ADSM server database and storage pool backup facilities. Incremental backups are provided as well as a mechanism for offsite backups to aid in disaster recovery.

1.2.4 Application Client

The application client is a software application that runs on a workstation and uses the ADSM application programming interface (API) to back up, archive, restore, or retrieve objects from an ADSM server.

As shown in Figure 5, the application client program enables other IBM and non-IBM products to use the storage management services of ADSM. The application client allows applications to back up or archive valuable data in any format that an application programmer specifies.

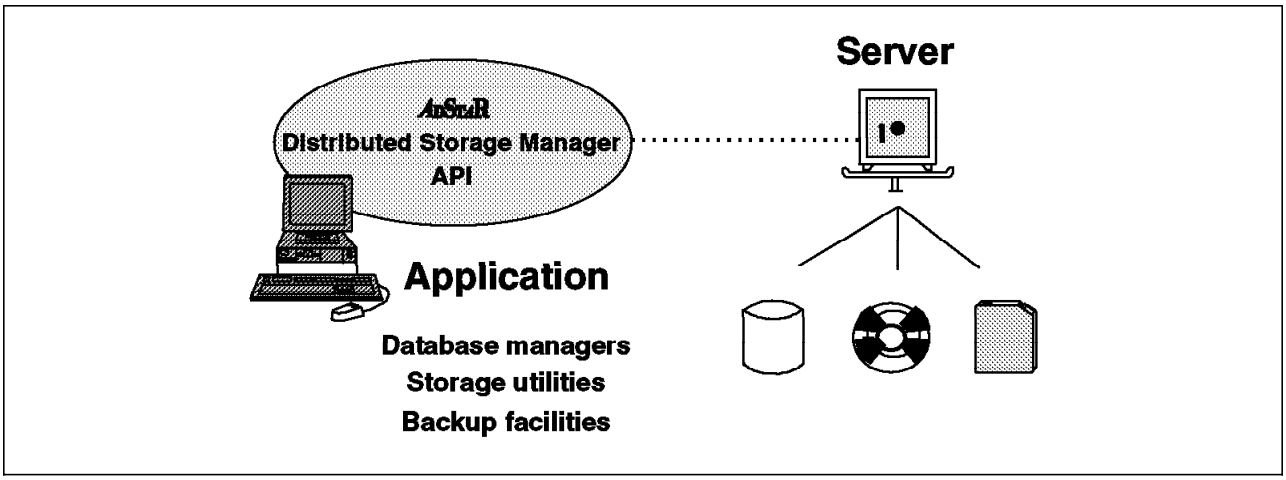


Figure 5. Application Client and ADSM API

The number of ways of using the API is unlimited. You can use it to provide better handling of nonfile data in the enterprise, such as databases or image volumes. You could, for example, provide extensions to the existing ADSM back up and restore functions to meet your user's needs or write a virtual tape device driver so that other applications can use ADSM transparently.

The API is available for the C programming language.

1.3 Functions

Let us look at the ADSM backup and restore, archive and retrieve, central scheduling, and policy management functions.

1.3.1 Backup and Restore

The backup process creates a copy of a client file on the ADSM server, such as the \myfile.data file shown in Figure 6. The backup process also backs up the directory in which the file resides.

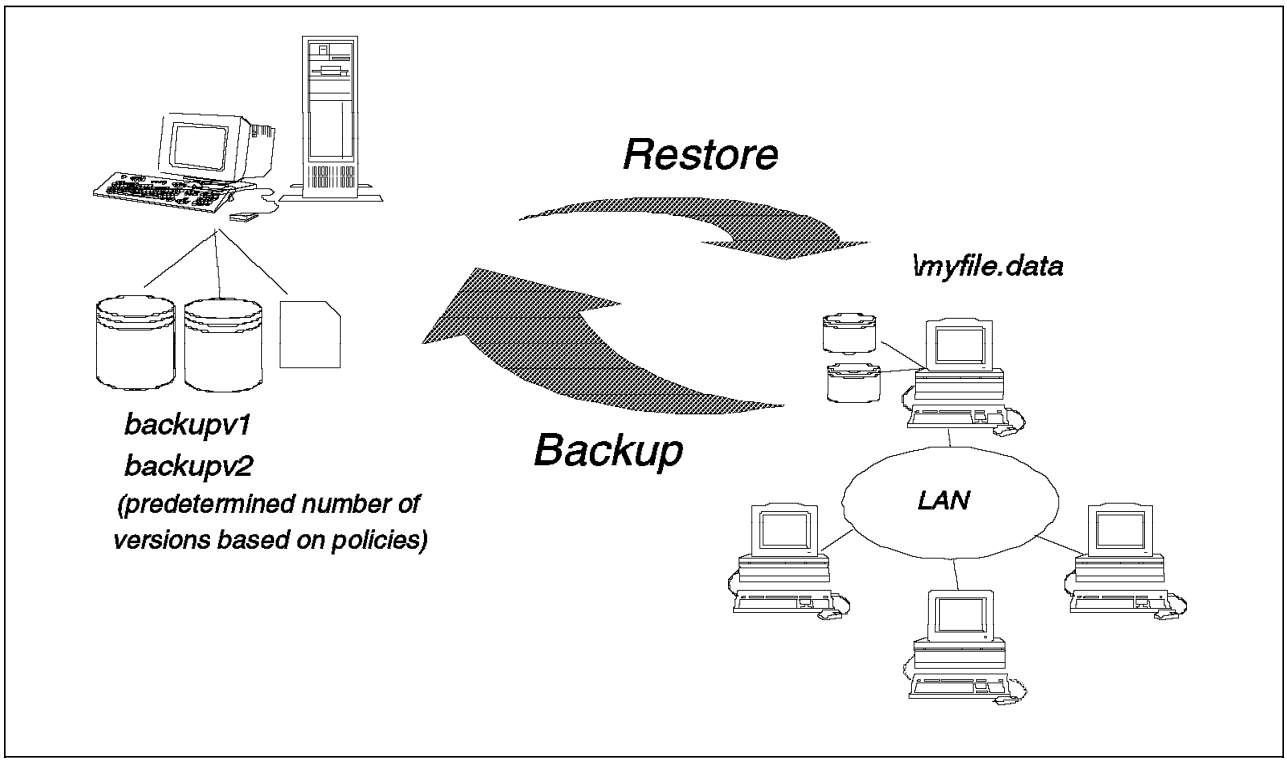


Figure 6. Backup and Restore

Incremental backup sends to the server the files that have changed since the last backup. The first time an incremental is done, all files are sent to the ADSM server. This is a full backup. ADSM determines that a file has changed if any of the following has changed: file size, date and/or time stamp, file owner, file group, file permission, or attribute change time.

Selective backup specifies which files a user wants to back up. A selective backup can consist of a single file, or a user can select a directory or subdirectory tree to back up. Because wildcards (*) are allowed in the specification, there is great flexibility in file selection.

The files are backed up according to policies that the administrator has predefined. The policies define, for example, how many backup versions should be retained in the ADSM storage pools, how long to retain those versions, and whether to back up files that are in use. Figure 6 shows that two versions of the \myfile.data file are saved in the storage pools.

Restore is the process of copying a backup version from the server to the client. This process is system assisted; that is, the system performs the restore for the user; the user does not have to call the ADSM administrator to request restoration of the file.

1.3.2 Archive and Retrieve

The archive process creates a copy of a client file on the ADSM server, such as the \myfile.data file shown in Figure 7.

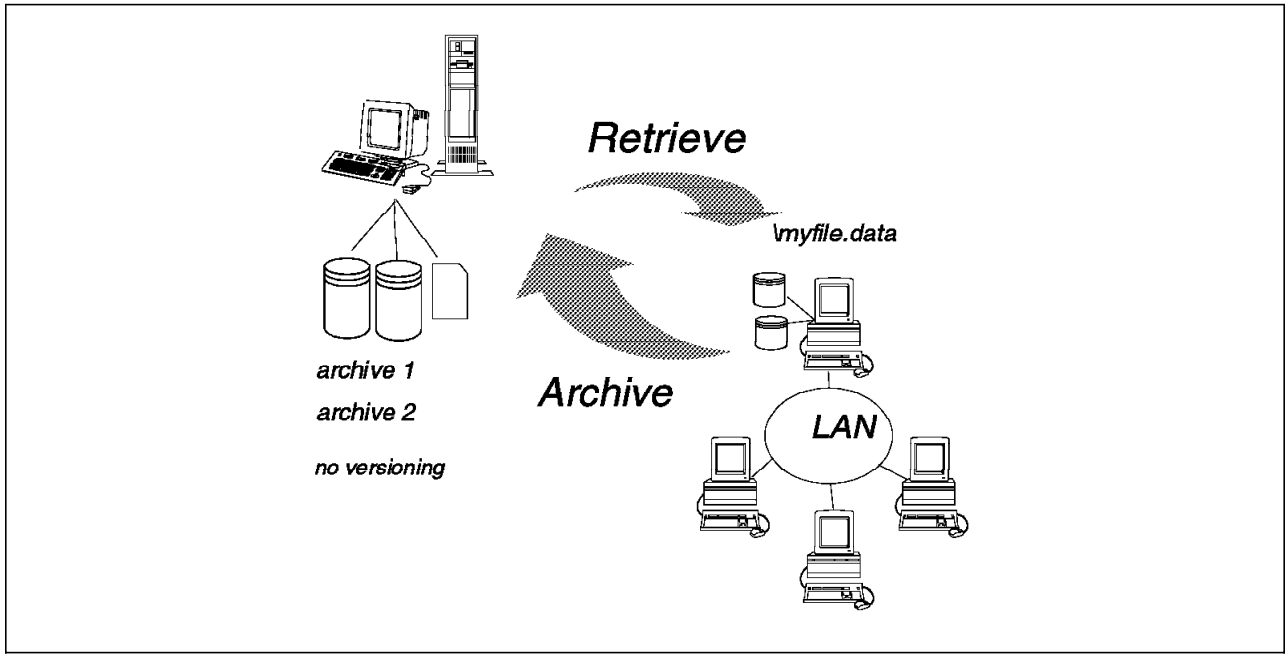


Figure 7. Archive and Retrieve

As with backup, archived files are managed on the basis of policies; however, the archive function does not have a concept of versioning. You can archive multiple versions of a file by invoking the archive function multiple times. In other words, each archived copy is treated as a separate file, not as multiple versions of a single file.

A user can save a description of an archived file so that it will be easier to retrieve the file if multiple files are archived with the same file name.

The key difference between backing up a file and archiving a file is that the user can erase the original file after archiving it. The archived version is expected to be retained for a long time. Erasing the original file does not affect the retention period for the archived file.

1.3.3 Central Scheduling

As shown in Figure 8 on page 12, the ADSM central scheduling facility automates the initiation of client backup, archive, restore, and retrieve, as well as ADSM server administrative operations. It also can schedule any client OS command and ADSM client macros. New clients can be easily associated with schedules in a nondisruptive manner. The central scheduler consists of client and server processes that cooperate to execute the scheduled functions. Thus ADSM requires the client workstations to be communicating with the server. If you want to automate your backups for off-hours or weekends, you must enforce a policy that requires users to leave their workstations powered on.

The administrator is responsible for defining and maintaining the schedules and has the authority to prioritize clients so that clients that contain more important data are given preferential treatment.

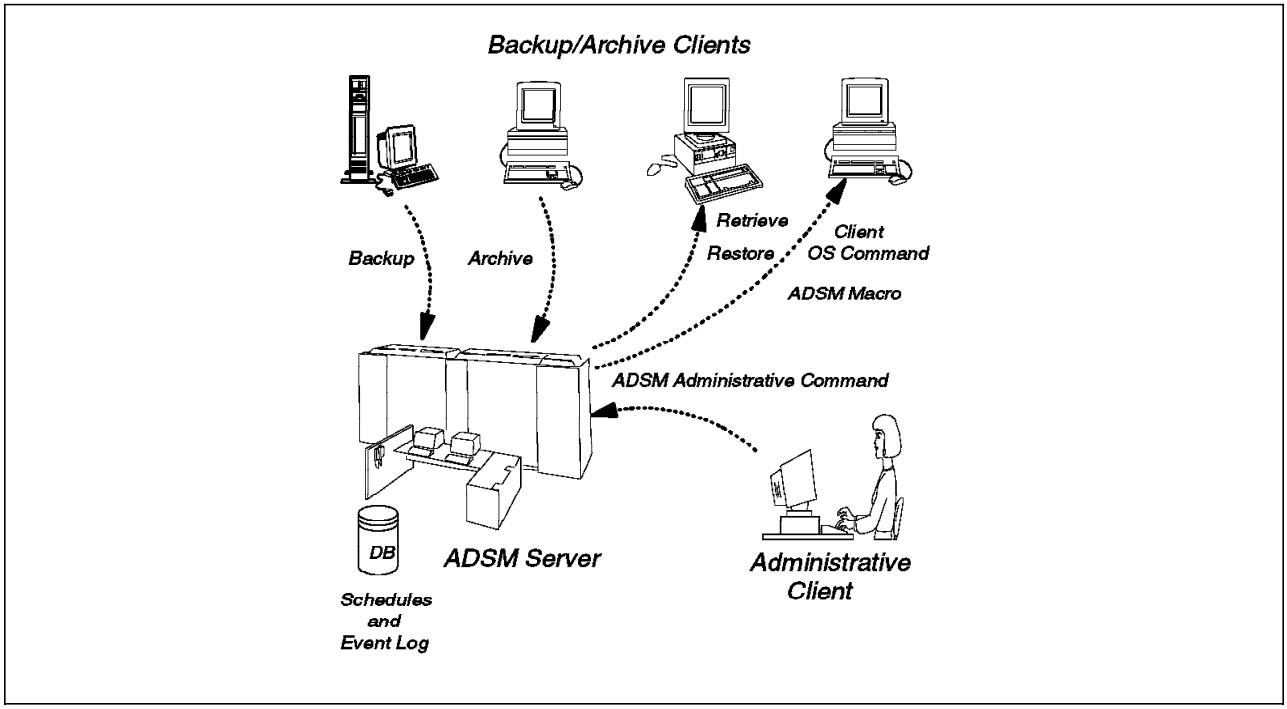


Figure 8. ADSM Central Scheduling

A schedule event log is maintained in the server database. Whenever a schedule process starts or fails, an event record is written to the log. An administrator can query the log to determine whether scheduled events completed successfully or not.

Two types of scheduling modes are supported:

- Client polling

  Client polling is supported for all client workstations and all communication methods. With client polling, a client periodically queries (or polls) the server for a scheduled operation and the date and time the operation is to start. The server sends this information to the client. Then the client waits until it is time to start the scheduled operation and executes the operations. Operations that can be scheduled are backup and archive. Restore and retrieve cannot be scheduled. Before executing the operation, the client notifies the server that a scheduled operation is starting. Upon completing the operation, the client notifies the server that the operation has completed either successfully or unsuccessfully.

  The client initiates client polling by starting the client scheduling program. To start the program the client enters DSMC SCHEDULE. The program will continue to query the server and execute schedules until the user explicitly stops the program or the machine is shut down.

- Server-prompted

  Server-prompted scheduling is supported for all client workstations that use TCP/IP to communicate with the server.

  With server-prompted scheduling, the ADSM client registers its TCP/IP address with the server and then waits to be prompted by the server to begin the scheduled operation. The client then starts and executes the operation. The operation can be backup or archive. Restores and retrieves

cannot be scheduled. Upon completing the operation, the client notifies the server that the operation completed either successfully or unsuccessfully.

Server-prompted scheduling allows the server to control when clients are contacted to perform a scheduled operation. Server-prompted scheduling maximizes the use of scheduled sessions.

To initiate a schedule, the client must start the client scheduling program by issuing the DSMC SCHEDULE command.

1.3.4 Policy Management

As shown in Figure 9, ADSM allows you to manage the backup and archive process based on policies you establish for your enterprise. The granularity of control that you have is down to the file level. You can decide on how granular you want your policies to be. You can establish an overall system policy, policies by department or organizational structure, or policies by user or file name. Policy management makes ADSM a true system-managed storage implementation. The elements of policy management are discussed below.

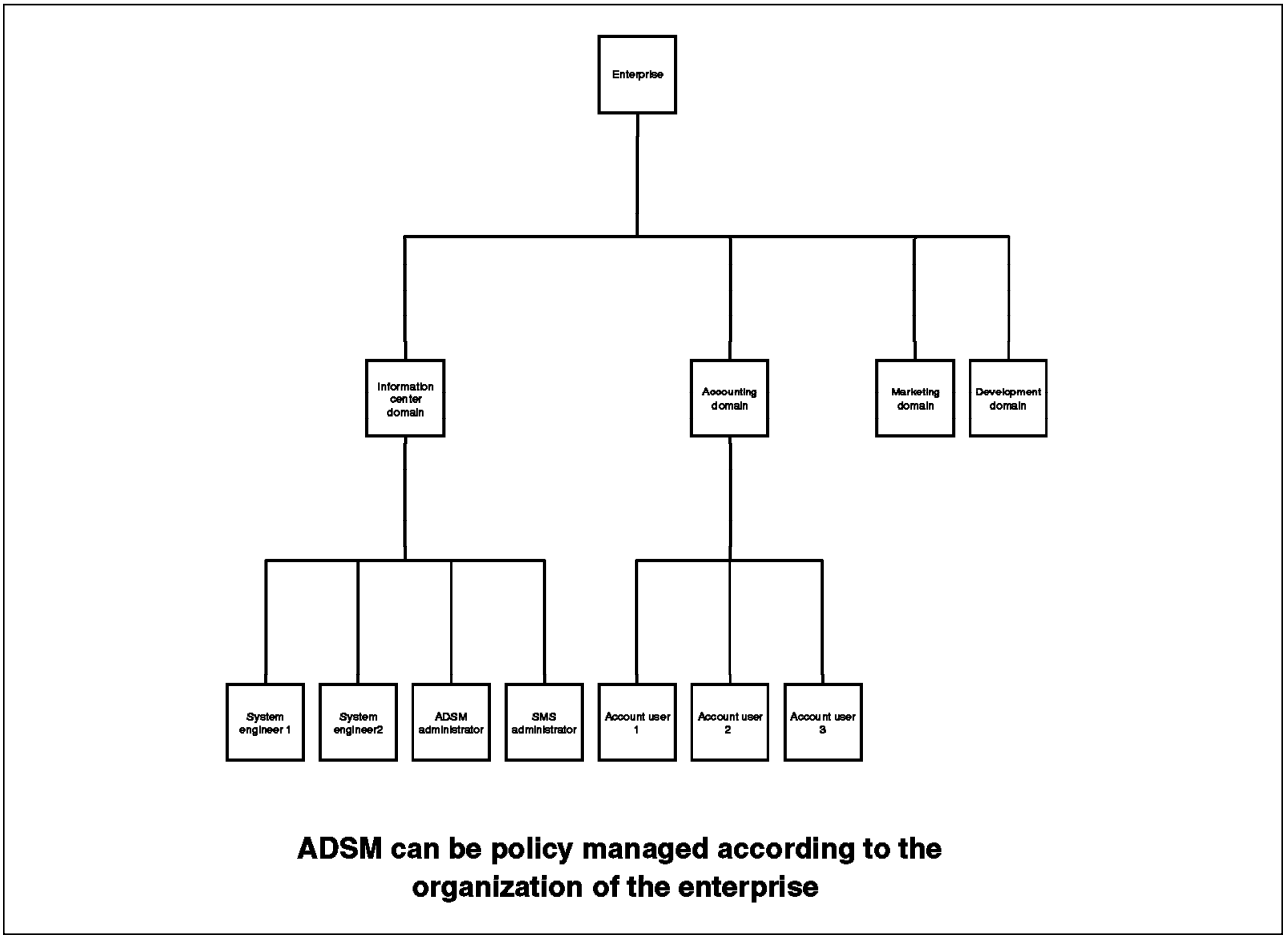


Figure 9. Policy Management

1.3.4.1 Policy Domain

A policy domain is a group of clients who are working according to the same set of policy needs. A policy domain provides a logical way of managing backup and archive policies for a group of client nodes. There is no limit to the number of policy domains that can be defined on an ADSM server. Policy domains can be used to provide standard storage management policies to most users, group together clients that have similar storage management requirements, limit the number of clients to be managed by a single policy administrator, or restrict the number of management classes to which users have access. Figure 10 on page 15 shows that clients ACCTREP1, ACCTREP2, and ACCTREP3 belong to the SALES policy domain. Note that schedules also belong to a particular policy domain.

1.3.4.2 Policy Set

Each policy domain can contain one or more policy sets. A policy set contains one or more management classes. A policy domain can have more than one policy set, but only one policy set can be activated at any point in time. Each policy set contains a default management class and can contain any number of additional management classes. Policy domain and policy set information is stored in the server database. Figure 10 on page 15 shows three policy sets with an active policy set that contains two management classes.

1.3.4.3 Management Class

Policy sets contain one or more management classes. Management classes contain a backup copy group and/or an archive copy group or no copy group. You can think of management classes as a Service Level Agreement you have with your clients on how their backup and archive data will be handled. There is a concept of binding the management class to the file when it is backed up or archived. Thus the management class is associated with that file. You can rebind a file with a new management class. Users can use the default management class or explicitly select a management class that is within the active policy set to which they have access. Figure 10 on page 15 shows two management classes, MC1 and MC2. Management class MC1 contains both a backup and archive copy group; MC2 contains only an archive copy group.

1.3.4.4 Copy Group

Copy groups are where you specify the parameters that will control the generation and expiration of backup and archive data. There is a separate copy group for backup and one for archive. In the current ADSM product, all copy groups are named STANDARD. Again, copy group information is stored in the ADSM server database. Let us look at the parameters that you can use to control your backup and archive data. Remember that the span of control is at a file level.

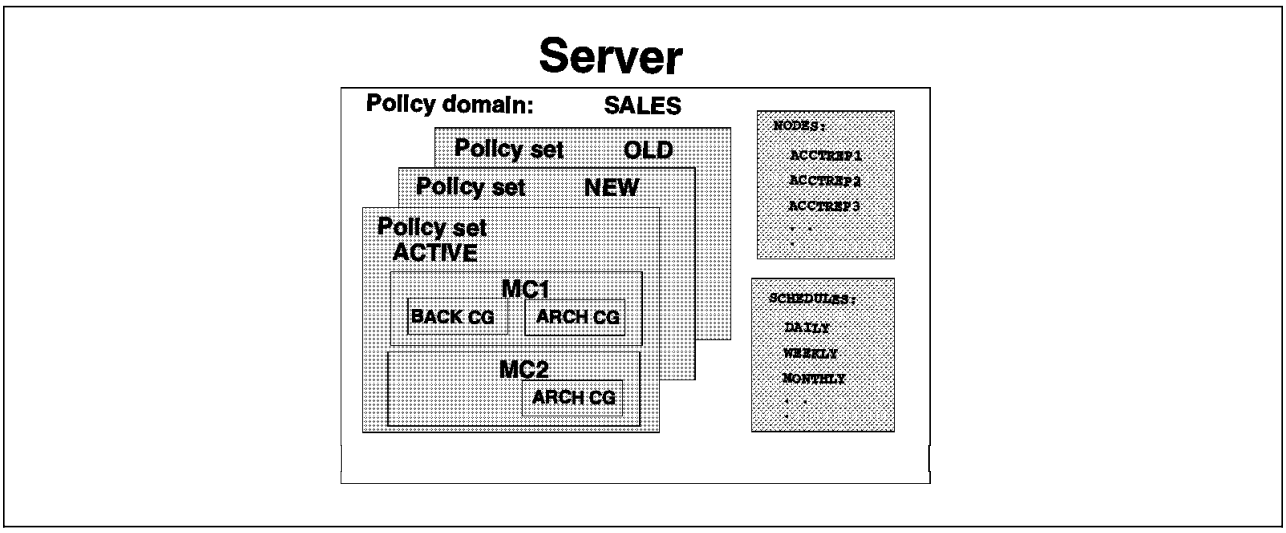


Figure 10. Policy Management Elements

Both backup and archive copy groups have similar parameters except that there is no concept of versioning with archived files. Let us look at each parameter shown in Figure 11.

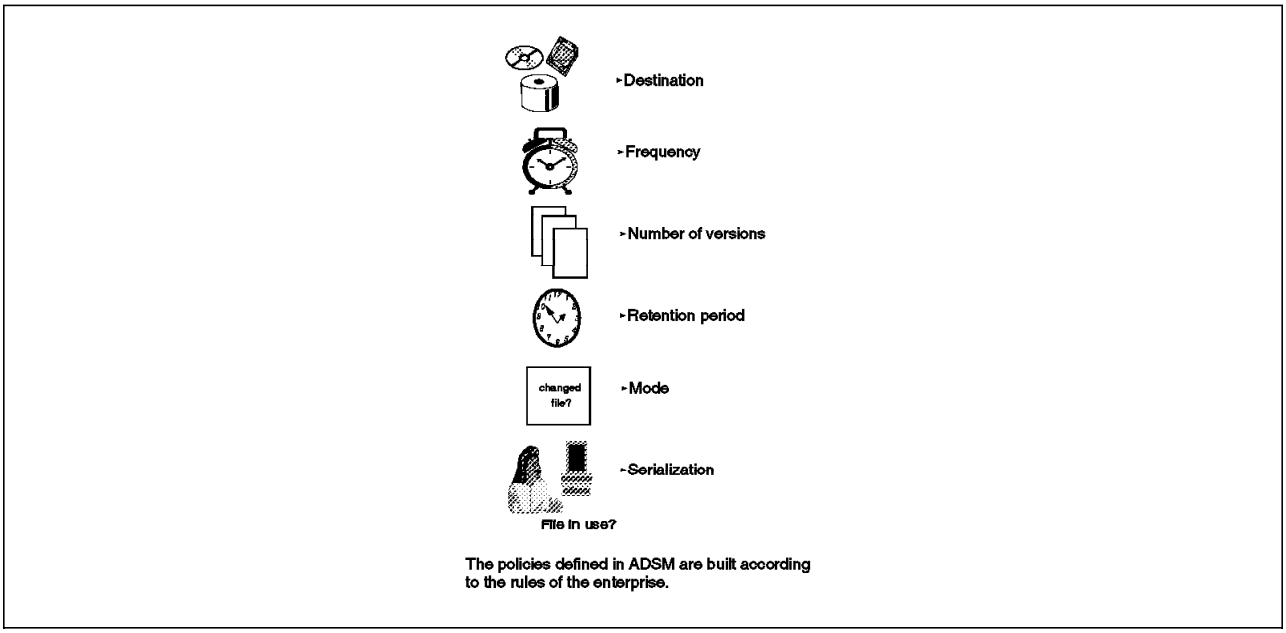


Figure 11. Copy Group Parameters

Destination specifies the name of the storage pool where the server stores the backed up or archived files.

Frequency for a backup file specifies the minimum number of days that must elapse between incremental backups. This parameter is not used for selective backups. Frequency for an archive file is always command (CMD). A file is only archived when a client issues an archive command or chooses archive from the GUI.

The concept of **versioning** applies only to backup files. You can specify two different parameters to tell ADSM how many versions of a backup file you want it to maintain. The Version data exists parameter specifies the maximum number

of different backup versions the server retains for files and directories that currently exist on the client workstation. The most current backup version is called the active version. All other versions are called the inactive versions. When the maximum number of versions is exceeded, the server rolls off the oldest version. The Version data deleted parameter specifies the maximum number of different backup versions the server retains for files and directories that have been erased from the client workstation.

The **retention period** parameter specifies how long to retain the backed up and archived files. There are two retention parameters for backed up files that correspond to the two types of versioning, and there is one retention parameter for the archived files. Retain extra versions specifies how many days the server retains the inactive backup versions when the original file no longer exists on the client′s workstation. Retain only version specifies how many days the server retains the backup versions it has of a file when the original file has been deleted from the workstation. Retain version specifies the number of days an archived copy remains in data storage.

With the **mode** parameter you can specify file backup depending on whether the file has changed since the last backup. This parameter applies to incremental backups, not selective backups. The options for mode are modified and absolute. Modified means that you want to back up the file only if it has changed. Absolute means that you want to back up the file regardless of whether it has changed. For archive files, the mode is always absolute.

Serialization specifies how files or directories are handled if they are modified during the backup or archive process. The serialization parameter has four options: static, shared static, shared dynamic, and dynamic:

- **Static** specifies that if a file or directory is modified during the backup or archive process, ADSM will not back up or archive the file. The static mode is not supported on the DOS platform.
- **Shared static** specifies that ADSM will retry the backup operation as many times as specified in the client′s option file. The default is four retries. If the file or directory is modified during each backup or archive attempt, ADSM will not back up or archive the file.
- **Shared dynamic** specifies that if a file is modified during a backup or archive attempt, ADSM will only back up or archive the file on its last retry.
- **Dynamic** specifies that even if the file is modified during the backup or archive attempt, ADSM will back up or archive the file anyway. No retries are required.

1.4 Advantages

IBM′s commitment to provide storage management functions for remote systems and multiple platforms will enable the enterprise to gain substantial benefits. Let us conclude this section of the book by summarizing the many advantages of ADSM. We have grouped them into three categories: cost reductions, increased productivity, and increased security of corporate assets.

1.4.1 Cost Reductions

As proven in the large systems arena, fewer people will need to be involved in the process of storage management. Because ADSM provides a single approach to storage management for your entire enterprise, perhaps all enterprise storage can be managed by one central group.

Available storage is managed more effectively because inactive data can be moved to other media, thus increasing the amount of fast access disk space required for active data.

Distributed data growth can be easily monitored and controlled because less duplication of data is likely to occur as users become confident that their data is secured. ADSM can also help you avoid backing up multiple copies of system software.

1.4.2 Increased Productivity

With storage management across an enterprise, any issues regarding data integrity can be restored by the system more quickly and accurately while adhering to enterprise-defined standards.

Many of today′s tasks that many people do manually can now be automatically managed by fewer people. End users or system administrators from different systems and platforms will spend less time on availability and space management. They will experience fewer losses of critical data and fewer out-of-space incidents. They will be freed from the tedious manual work of managing the disks they own; data management can be done transparently with little impact on their usual work.

Using a single storage management strategy across your heterogeneous platforms encourages networkwide standards. This might be at a high level, where standard policies of how to manage the data are established, or at a lower level, where, for example, file naming conventions become more standardized to improve restorability to other platforms.

1.4.3 Increased Security of Corporate Assets

With a structured process for backing up your data in your enterprise, you can be ensured that your data will be available when you need it. The risk of losing valuable data, which is a valuable corporate asset, has been substantially reduced.

You may also be taking advantage of more reliable server storage than that provided on your local workstation. The system on which the server storage resides may also have a higher level of security enforcement and control. The server storage might provide a higher level of security simply by being in a central data center (glass house) that requires a badge lock to enter.

Now that you know what ADSM is all about, let us look at how you can use ADSM to back up and manage your Lotus Notes data including databases and system configuration information.

Chapter 2. Relational Databases

This chapter contains an overview of relational databases and explains why they have become so popular. A section on terminology is included to help avoid confusion with differences in terminology among RDBMS products. The fundamental structures, such as tables, table spaces, data files, control files, parameter files, and configuration files, are described. Specific data storage considerations for UNIX-oriented systems, such as using raw devices or file system files, are discussed.

2.1 Why Relational Database?

Relational database management systems (RDBMSs) have become so popular since they became widely available in the early 1980s that most line-of-business applications that are being implemented today use them. Some people predict that by the end of the century relational databases will have been superseded by object databases. Whether or not this prediction turns out to be accurate is not of immediate importance to those of us who have the task of backing up the data used by today′s applications.

The purpose of this section is to provide a brief explanation of why relational databases have become so popular. It answers two questions: Why database? and Why relational?

Let us first look at the history of database systems. In the early days of commercial computing (in the 1950s and 1960s) programmers included in the body of their programs the code to read and write to peripheral devices. As peripheral devices became more sophisticated and especially when direct access storage devices (DASDs) became available, input and output became more complicated. As a consequence, vendors started to provide standard routines for handling input and output. These came to be known as access methods.

The main benefit to users of access methods was that application programmers could be more productive. The benefit to computer vendors was that access methods removed a major inhibitor to the sale of new peripheral devices—the need to make changes to application programs. As long as a program′s interface to the access method remained unchanged, the vendor′s customers could buy and install new equipment (together with associated access methods) secure in the knowledge that their application programs would continue to work.

Over time most customers developed new applications. Some of these new applications required access to the same data as other applications. This presented application designers with two options. One option was to build a new set of files and manage and maintain them independently of other applications. The other option was to find a way of using the files that already existed and were used, managed, and maintained by existing applications.

The major disadvantage of the first option was that it led to a proliferation of files, many of which duplicated the data held by others. This proliferation led to problems in ensuring that all of the files were consistent with each other. It also led to wasteful use of storage and processing power because the same information needed to be held and maintained many times.

However, the second option often required changes to file structures. Changes to file structures meant changes to application programs, and it was usually difficult and time-consuming to modify and test all of the application programs affected by even a simple change to a file structure.

Database management systems (DBMSs) were developed to combat these problems. Specific benefits of implementing a database management system were that an organization was able to:

- Store data only once
- Ensure that data was consistent
- Allow many users and applications to share the data
- Install new storage devices and change data structures without making changes to application programs.

Let us now look at the history of RDBMSs. The first widely available DBMSs were based on hierarchical or network structures. They were record-oriented and adopted a navigational approach to database processing.

Initially these structures satisfied the requirements of businesses and their users. However, as the databases and their use grew, so the task of managing them became harder. Although several applications needed access to the same data, often they needed to view the data in different ways. Database administrators (DBAs) developed structures to provide these different views, but often the result was ever-increasing complexity. Users also found these hierarchical and network structures difficult to understand and deal with. Users were required to keep track of their current position in the database, and relationships between data were visibly implemented by pointers.

It became apparent that, although existing DBMSs had solved many problems, they had created some new ones.

RDBMSs were developed to address both the old and new problems. They provide all of the benefits of other DBMSs, but instead of holding their data in the form of hierarchies or networks they hold it in the form of two-dimensional tables. These tables (sometimes also referred to as relations) are similar to sequential files; however, a number of rules that relational theory imposes on these tables are not always applied to sequential files. These rules are:

- Each table contains only one type of record (row).
- Each row contains the same number of fields (columns).
- The order in which rows appear in a table is unspecified.

Figure 12 on page 21 is an example of a table, called ITSC Friends, which contains five rows, each comprised of four columns. These two-dimensional tables are easy for users to understand and manipulate. They also allow different users and applications to view and process the same data in different ways without requiring complex structures.

ITSC Friends

serial number	name	country	product area
12345	Joshua Peleg	Israel	ADSM
98765	Craig Welch	USA	VM
13579	Guido DeSimoni	Italy	CICS/6000
24680	Hugh Smith	United Kingdom	CICS/MVS
10293	Sally Montera	USA	MVS

Figure 12. Example of a Table (Relation)

The popularity of RDBMSs stems from the fact that they provide all of the benefits of other DBMSs but are simpler and more flexible to use and to manage and have a sound theoretical basis in mathematical relational theory.

2.2 Terminology Used in This Book

Although all RDBMSs are based on the same set of principles, they do not all use the same terminology. For example, Oracle's *table space* is the equivalent of INFORMIX-OnLine's *dbspace*. Other terms are used so loosely as to be ambiguous and open to misinterpretation. The term *archive* is one such example.

This section defines terms as they are used in the general sections of this book. Sections that focus on a specific RDBMS use the terminology of that RDBMS product.

Table space A table space provides the link between the logical view of a database that the user sees and the data files that the database uses to hold the data. The relationships between table spaces, tables, and data files are described in 2.4, "Fundamentals of Relational Database Management Systems" on page 25.

Equivalents in the different RDBMSs are:

- Oracle - *table space*
- INFORMIX-OnLine - *dbspace*
- Sybase- *segment*
- DATABASE 2 AIX/6000 - *database*
- DB2/2 Version 2 - *database*.

Archive In the world of computing that exists outside this book the term *archive* has come to mean many different things. Sometimes the term is used as an equivalent to *backup*. Sometimes it is used to describe the process by which a data file is migrated through a storage hierarchy.

In this book the term *archive* is used to mean the technique of storing away a set of objects that you no longer require for regular use but may need to retrieve at some future time. Once you have *archived* objects, you usually delete them from disk and release the space that they occupy for other use.

Restore The term *restore* is used in this book to mean the act of restoring data from a backup copy. It specifically excludes the process of bringing the data up to date by applying log files.

Recover The term *recover* is used to include the processes of restoring data and bringing the data up to date by applying log files.

Retrieve The term *retrieve* is used only in connection with *archived* objects. It describes the process of copying archived objects into online storage so that users or application programs can work on them.

Load Most RDBMSs provide a *load* facility. The purpose of this facility is to allow large volumes of data to be inserted into a table quickly and efficiently. The facility usually supports different techniques for splitting the data up into columns. For example, you may specify a separator character (such as a comma) to delimit input data that is to be loaded into a particular column. Alternatively, you may specify start and end positions of input data.

Export and Import Most RDBMSs provide a pair of utilities that, in this book, we refer to as export and import.

The export utility writes out to a flat file the data (and the data definition) of a set of RDBMS objects. This set of objects may be a single table. Alternatively, it may be an entire database.

The import utility usually works only on files that have been created previously by the export utility. These utilities are often used for moving data tables from one database to another, or as part of the process of migrating to a new version of the RDBMS.

With regard to backup and recovery, the export and import utilities are sometimes useful in that they enable you to convert database objects into a form that can be processed by other products (ADSM, for example.), or they may be the only way to back up data at a table space level. Some RDBMSs provide more than two utilities to supply the functions just described. For example, INFORMIX-OnLine provides dbexport, dbimport, tbunload, and tbload.

Physical Backup A physical backup of a database is defined as a type of backup where you save one or more files as part of the database structure. If you back up a database with ADSM commands directly, this is referred as a type of physical backup because you back up the physical structure of a database (from the operating systems point of view) regardless of their logcal contents.

Logical Backup A logical backup of a database unloads (exports) the contents of a database to file system files. An export operation is referred as a type of logical backup where the RDBMS product creates

file system files to store the logical units of the database (records) regardless their physical structure.

Log file RDBMSs use *log files* to record the changes made to databases. Log files often can be used to maintain database consistency in the event of an error or failure. Different RDBMS suppliers use different terms for log files. Equivalents in the different RDBMSs are:

- Oracle - *Redo logs*
- INFORMIX-OnLine - *Logical logs*
- Sybase - *Transaction logs*
- DATABASE 2 AIX/6000 - *Log files*
- DB2/2 Version 2 - *Log files*.

Raw devices AIX for RISC System/6000 supports two techniques for storing data on disk, *raw devices* and *Journaled File System*. All of the UNIX-oriented RDBMSs covered by this book allow you to use both techniques. Other terms that are often used to mean the same thing as *raw device* are *raw partition* and *logical volume*.

Journaled File System The Journaled File System is the alternative to raw device support that AIX for RISC System/6000 provides for storing data on disk. The most conspicuous difference between raw device files and Journaled File Systems is that raw devices can be defined only in the /dev directory, whereas the Journaled File Systems can be defined in any directory other than /dev. Another difference is that any structure that exists within a raw device is invisible to the user, whereas you usually can see the contents and structure of a Journaled File System. (The structure takes the form of subdirectories and files.) I/O to the JFS is controlled by the operating system, whereas I/O to a raw device is controlled mainly by the application.

file system file The term *file system file* is used in this book to refer to data files that form part of a Journaled File System. Sometimes these files are called *cooked files* or *flat ascii files*.

Logical volumes The Logical Volume Manager (LVM) is an important feature of AIX for RISC System/6000. It allows you to manage one physical disk volume as though it were several different disk volumes. The LVM also allows you to manage several physical disk volumes as though they were a single volume. You can achieve this sleight of hand by defining a logical volume to the LVM. When you define a logical volume, you specify the physical disk volumes that you want to use and how much space you require. A logical volume may be used either as a raw device or to hold Journaled File System files.

Data file The term *data file* is used to refer to those UNIX files that RDBMSs use for storing data. Data files may be either Journaled File System files or raw devices. INFORMIX-OnLine uses the term *chunks* to refer to data files that make up table spaces (or *dbspaces* as they are known to INFORMIX-OnLine.)

2.3 Fundamentals of Data Storage in AIX

An AIX application performs input/output operations to a disk in one of two ways. The first way is to use AIX for RISC System/6000′s Journaled File System. The second way is for the application to use its own input/output routines, bypassing most of the routines supplied as part of AIX for RISC System/6000 and dealing directly with the disk devices. In this book we use the term “raw device” to refer to this second way.

2.3.1 Journaled File System

In general, the benefits of using the Journaled File System are as follows:

- File system integrity changes are journalled.

  AIX for RISC System/6000 records changes that are made to Journaled File System files. Therefore if an input/output operation is interrupted by a power failure (or by the user switching off the RISC System/6000), AIX for RISC System/6000 will, when restarted, recover Journaled File System file system structure to a consistent state. This does not guarantee that the data within the file will be correct but it does avoid the whole file system becoming corrupted and unusable.

- Data is easy to manage.

  As shown in Figure 13, Journaled File System files exist in hierarchical structures of files and directories. Users can use AIX commands and utilities to explore these hierarchies. Users can also change these hierarchies, adding new directories and removing others, or moving files from one directory to another.

  The structure of data held on raw devices is known only to the application that wrote the data. Some utilities can operate on data held in raw devices (for example, the *dd* command). However, these commands deal with the entire raw device as a single unit of data. None of the utilities supplied by AIX for RISC System/6000 can see the structure of the data that is held on raw devices.

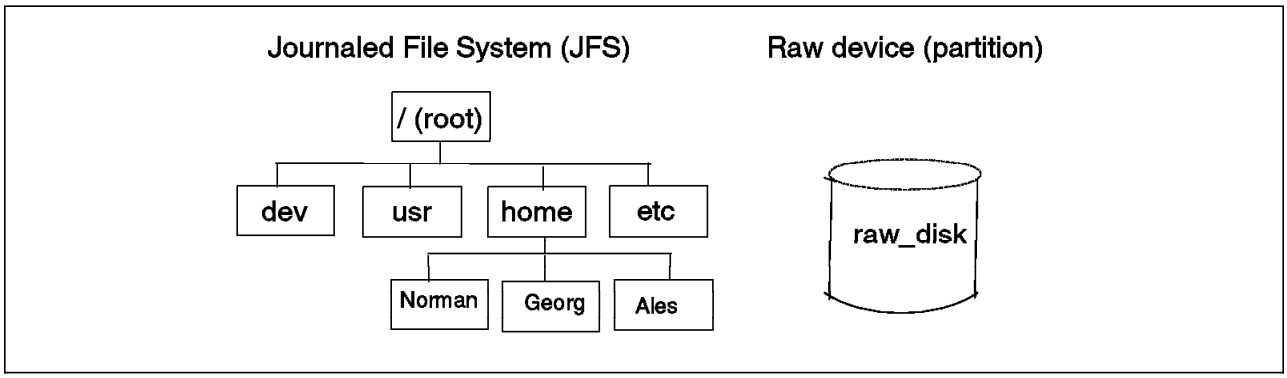


Figure 13. JFS and Raw Devices

- The Journaled File System provides good performance.

  The AIX for RISC System/6000 Journaled File System is capable of operating very efficiently when running on RISC System/6000s. Programs that take advantage of this efficiency can often provide better performance than similar programs that operate on raw devices.

A limitation imposed by Version 3 and below of AIX for RISC System/6000 is that no Journaled File System file can be larger that 2GB in size. There is no such restriction on raw devices. This restriction on the size of Journaled File System files may at first seem to be a serious problem for database designers because some databases need to be considerably larger than 2GB. However, RDBMSs allow you to build a database out of multiple files, and so the limitation on the size of Journaled File System files turns out not to be a problem after all.

2.3.2 Raw Devices

In general, the benefits of data held on raw devices are as follows:

- Raw devices offer better security than Journaled File System files.

  Only the application that created the data on a raw device knows how to read the data. This makes it difficult for unauthorized users, applications, and utilities to process data held on raw devices.

- Raw devices provide good performance.

  When an application uses raw devices for its input and output operations, the application has control over the efficiency of access to data on disk. Some applications run more efficiently when operating on raw devices. However, this is not universally true, and some applications work more efficiently when they use the Journaled File System.

2.3.3 RDBMSs and AIX Data Storage

All of the AIX-based RDBMSs that we consider in this book can use both Journaled File System files and raw devices. The database administrator chooses which method to use for each database. Therefore, this book explores backup techniques for both methods.

2.4 Fundamentals of Relational Database Management Systems

RDBMSs share a common set of principles. The purpose of this section is to explain a subset of these principles that a designer needs to understand in order to design a backup and recovery system for data held on a relational database.

2.4.1 Tables, Table Spaces, and Physical Files

You may find Figure 14 on page 26 and Figure 15 on page 26 useful aids to understanding how the various logical and physical RDBMS constructs interrelate.

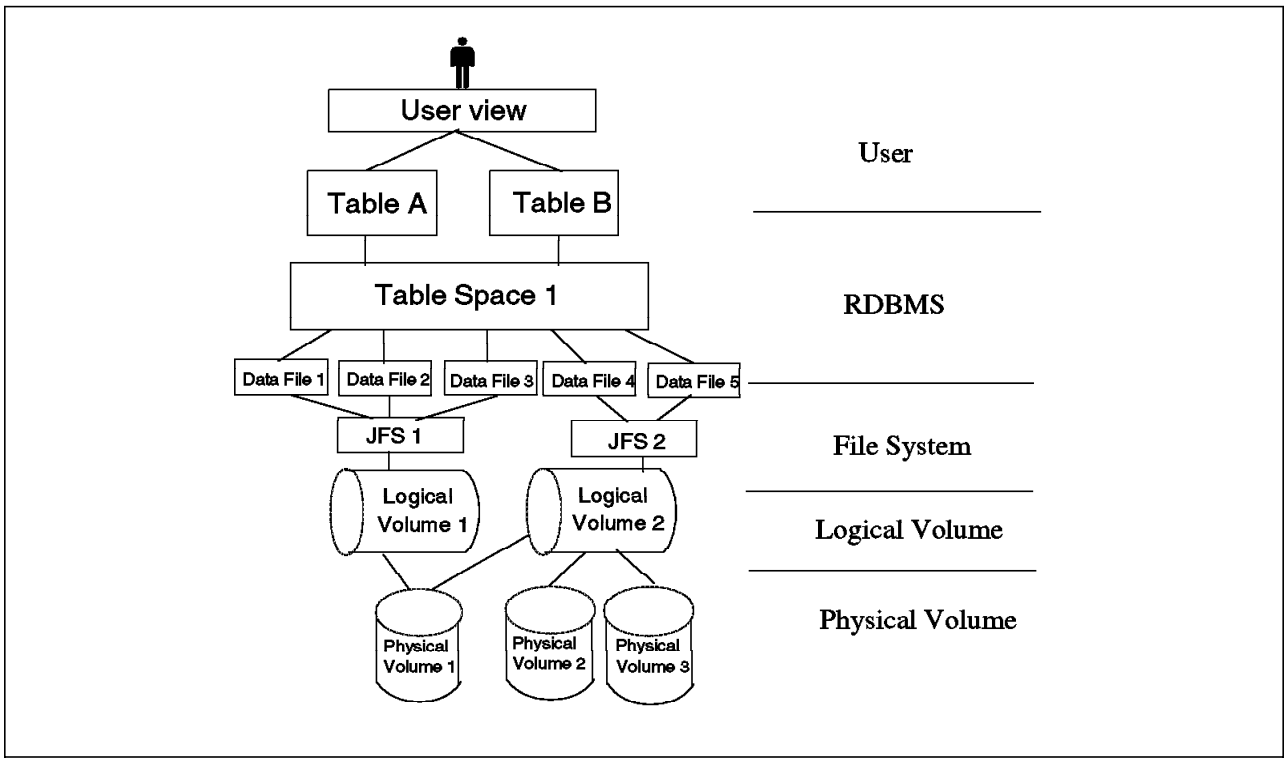


Figure 14. How RDBMS Tables Are Implemented Using JFS Files

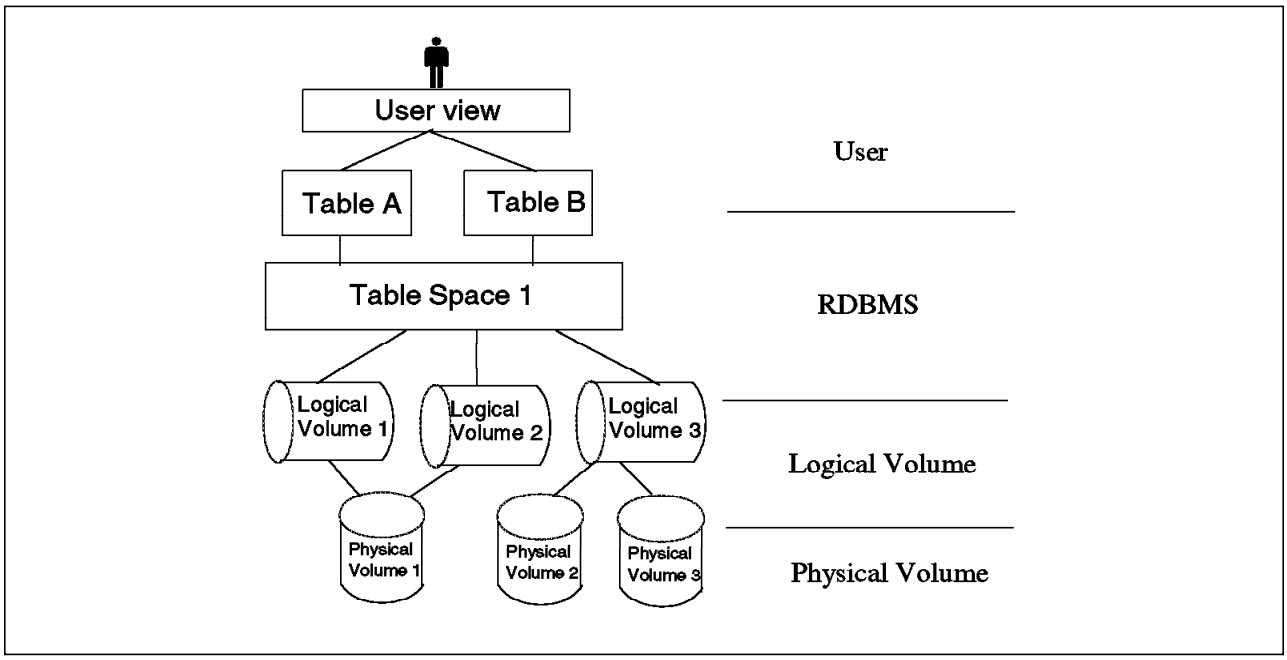


Figure 15. How RDBMS Tables Are Implemented Using Raw Devices

The user's view is that data is held in a number of two-dimensioanl tables. In Figure 14 and Figure 15, Table A and Table B are examples of these two-dimensional tables.

When users create a table they create it within a table space. They may specify the name of the table space they want to use, or the RDBMS may create the table in the user's default table space.

Table spaces are logical concepts RDBMSs use. They provide a convenient way of separating the user's view of data from some of the practical considerations associated with storing that data on disk. For example, a database administrator can make more disk space available to several tables by adding disk space to the appropriate table space, thus ensuring that tables do not run out of space and that disk space is used efficiently. Furthermore, the table space concept means that neither users nor application programs need to be aware of the fact that the database administrator has made more disk space available.

In an AIX environment table spaces can be implemented using either Journaled File System files or raw devices. Figure 14 on page 26 and Figure 15 on page 26 depict these two implementations.

In Figure 14 on page 26, the user's logical view of the data is represented by the two tables, Table A and Table B. The RDBMS holds the data for these tables in Journaled File System files represented by Data Files 1 to 5.

Table Space 1 provides the link between the logical views and the Journaled File System files. Points to note are:

- The data for Table A may be contained in only one of the data files.
- Alternatively, the data for Table A may be spread over two or all three of the data files.
- Each of the data files may contain data for both Table A and Table B.
- Each data file may reside in a separate JFS.

The significance of these alternatives is that the only way to back up or recover individual tables is by using the facilities that the RDBMS provides. However, data files cannot be shared by multiple table spaces. In other words, once the link has been established between a table space and a data file, the table space will have exclusive use of that data file. This means that you can back up a table space by backing up the physical data files that it uses. Please note that there is a one-to-one mapping between a JFS and a logical volume. All files and directories in a JFS have to be located on the same logical volume, but a logical volume can map onto many physical volumes.

Now let us look at Figure 15 on page 26. The user's logical view of the data is represented by Table A and Table B. The RDBMS holds the data for these tables as raw devices in Logical Volume 1, Logical Volume 2, and Logical Volume 3.

Table Space 1 provides the link between the logical views and the logical volumes. Points to note are:

- It is possible for the data for Table A to be contained in a single Logical Volume.
- Alternatively, the data for Table A may be spread over two or all three of the logical volumes.
- Each of the logical volumes may contain data for both Table A and Table B.

The significance of these alternatives is that the only way to back up or recover individual tables is by using the facilities that the RDBMS provides. Usually, you would define each table space on a separate logical volume, although this is not a requirement for all RDBMSs. For example, INFORMIX-OnLine allows multiple table spaces on a single logical volume. If you do define each table space on a

separate logical volume, you can back up a table space by backing up the logical volumes that it uses.

2.4.2 Log Files

As shown in Figure 16, most RDBMSs maintain details of updates to databases in *log files*. If, for some reason, a transaction that makes a change to the database fails to complete successfully, the RDBMS′ recovery procedure will use the log file to detect that an update may be only partially complete and to undo any changes that the transaction had made to the database.

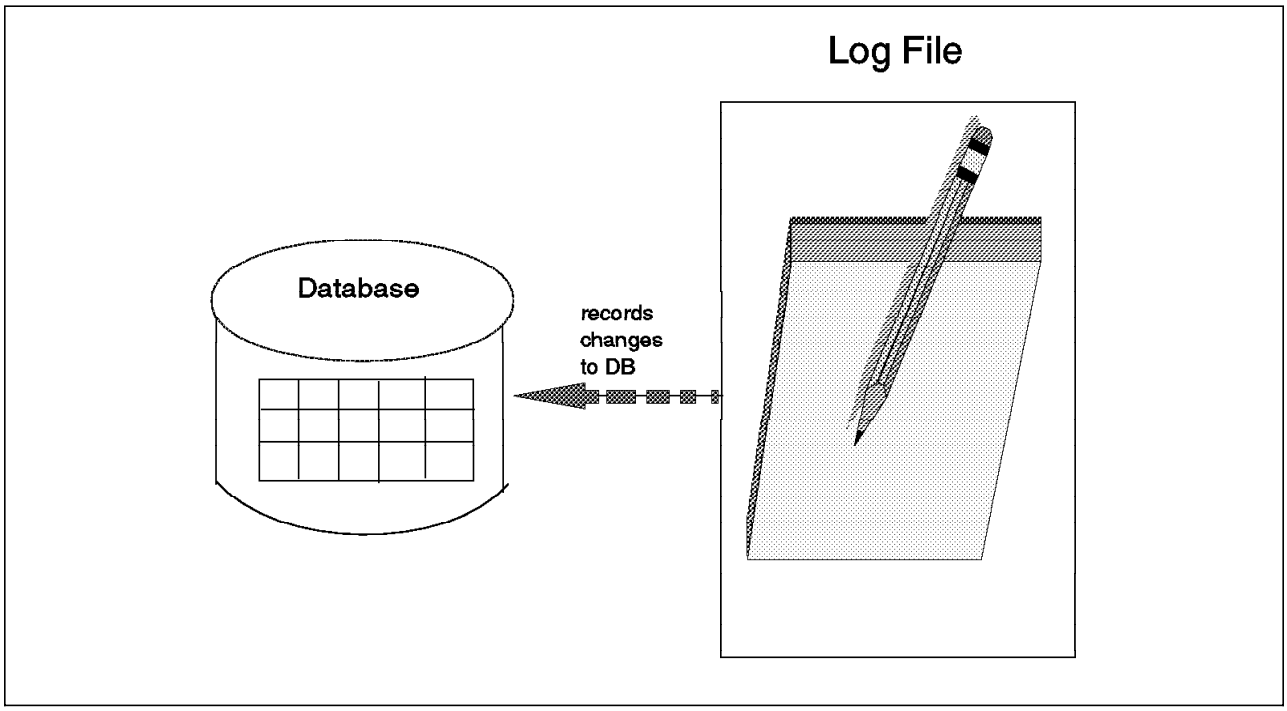


Figure 16. Log Files

Some RDBMSs support the use of log files to perform forward recovery. Forward recovery takes advantage of the fact that log files hold details of all changes that have been made to the database, and therefore you do not necessarily need to undo changes but instead can reapply changes. With forward recovery, the recovery process can:

- Restore a database to the state it was in at the time the last backup was taken
- Use the log files to reapply the changes that had been made since the last backup was taken
- Back out (undo) any partially completed changes.

A standard RDBMS concept related to log files is the checkpoint process. All RDBMSs use buffers in memory to hold changes to the database and log files. The purpose of buffers is to improve the operational performance of the RDBMS. However, the use of buffers means that most changes to databases and log files do not get written to disk until some time after the RDBMS has indicated to the user application that the update has been made successfully.

Checkpoints ensure that all database and log file changes held in the RDBMS′ buffers are flushed out to disk. This shortens the time it takes to recover a

database after a system crash because the number of redundant log records processed during the recovery is reduced. All RDBMSs support checkpoints and issue them automatically at intervals.

2.4.3 Control Files

Each RDBMS needs to hold information about the physical structure of the database, such as which physical files are used by each table space and which is the current log file. We call this information *control data*. Some RDBMSs (for example, Oracle) hold this data in separate files. Others (for example, INFORMIX-OnLine) hold it within the database in the table spaces.

We use the term *control files* to refer to files that hold control data. For those RDBMSs that hold control data in separate files, you need to define policies for backing up those files.

2.4.4 Initialization Parameter and Configuration Files

All RDBMSs provide a range of options. Some are set permanently, and others can be modified even when a database is in use (running). Some options allow you to tune the performance of the database; others allow you to specify how you want logging, for example, to be implemented. It is often more convenient to specify the options you require when you start up the database. All of the RDBMSs allow you to specify (at database startup time) a file that contains a list of how you want these options set initially. In this book we call these files *initialization parameter files*. Sometimes we abbreviate and call them *initialization files* or *parameter files*.

Installations may have multiple databases, and these databases may have multiple initialization parameter files. One reason for having multiple initialization parameter files for a single database might be to optimize performance for different circumstances. For example, you may decide to allocate one set of values when the database is used for batch processing and another set when it is used for online transactions. Although some of the options are set differently for each situation, many will be the same. Some RDBMSs allow you to specify options that are common to multiple initialization parameter files in *configuration files*. Instead of repeating all options and their values in each of the initialization parameter files you can select the configuration file that contains the options that you want to use.

You need to define policies for backing up both initialization parameter files and configuration files.

Chapter 3. Relational Database Backup

This chapter provides a general overview of how you can back up RDBMSs. We first discuss the requirements you need to factor in to your backup strategy, such as the types of errors you will recover from, how fast you need to recover, the backup window you have, and what you will actually recover. Then we describe the different techniques you can use for RDBMS backup, such as offline, online, export, full backups, partial backups, and backing up the log files. How ADSM applies to these techniques and how each RDBMS product supports these techniques is described in Chapter 4, “Exploiting ADSM” on page 49 and in Chapters 5 through 9 which describe specific database products.

We also look at some additional considerations you need to factor in to your backup strategy, such as the media you will back up to, the need to break up units of recovery, and whether you need to restore individual tables. A checklist is then provided to help guide you in defining your strategy.

We conclude with some backup considerations for advanced database implementations, such as handling very large databases, multiple database machines on the same or different machine, databases which are pointers to file system files, and archiving databases.

3.1 Backup Requirements

A backup strategy is only one part of your overall data management plan. You must consider how important your data is to the function (or even existance) of your organisation. The less time that your organisation can function without out its data the more important that data is to you. Your system must be designed in such a way to keep important data available when a failure occurs. Reliance on backups is not necessarily sufficient. You must also consider the following:

- Redundant Array of Inexpensive Disk (RAID) devices
- Dual access paths
- Dual I/O controllers
- Dual power supplies
- Backup or standby processors.
- Uninterruptable Power Supplies

None of these on their own can guarantee the availability of your data but in combination they can reduce the impact of a failure.

Before you can design a backup strategy you need to define the requirements that the strategy must satisfy. Factors that you will need to consider when defining the requirements for your backup strategy include:

Types of events The categories of incidents that may occur

Speed of recovery How quickly you need to be able to recover

Backup windows The periods of time at which backups can be performed

Recovery points To which points in time you need to be able to recover

Units of recovery Which other tables and files need to be recovered to the same point in time.

Let us look at each of these factors in more detail.

3.1.1 Types of Events

We identify five categories of events that may require data recovery:

- User error
- Statement failure
- Transaction failure
- Media failure
- Disaster.

Let us look at each category in more detail.

3.1.1.1 User Error

There is considerable opportunity for a user to make an error that causes data to be lost. For example, a user may inadvertently delete or update rows in a table or accidentally drop an entire table,or a programmer makes a logic error that results in data loss or corruption.

RDBMSs provide facilities that reduce the risk or impact of user errors. For example, you can use RDBMS security to restrict the data that individual users can access or update. However, it is not possible to eliminate the risk entirely, and you need to consider how to handle such situations.

One approach is to say that it is the user′s responsibility to recover from such errors. This approach may not be acceptable to users or their management, however. Another approach is to restore the entire database to the point in time at which the last backup was taken. This may not be satisfactory to other users who will lose the updates that they have made to the database since the last backup.

A third approach is to restore the table space that contains the damaged table. This approach is likely to be more acceptable than the other two because:

- It removes the responsibility for data recovery from the users.
- It may impact fewer users. The number of users impacted will depend partly on the number of tables included in the affected table space. (This topic is discussed in more detail in 4.3, “Additional Considerations” on page 56.)

You may, however, need to be able to restore individual tables, in which case you need to have backed up the tables individually.

3.1.1.2 Statement Failure

SQL statements that are syntactically correct may fail because, for example, the database is full. RDBMSs will usually detect such problems, roll back the effects of the failing statement, and report the problem to the user. Once the fundamental cause of the problem has been resolved, the user can retry the

statement and continue to work. There is normally no need to take any special action to recover from SQL statement failures.[1]

3.1.1.3 Transaction Failure

Transactions may fail for a variety of reasons:

- Programming errors
- Network failures
- Failures of the operating system or RDBMS
- Power failures.

The actions required to recover from these situations vary according to the particular circumstances. However, the RDBMS will ensure that the integrity of the data it manages is preserved. You do not need to restore data to recover from transaction failures.

3.1.1.4 Media Failure

RDBMSs normally use magnetic disk as the medium on which they store the data that they manage. If a disk volume is physically damaged or destroyed, at a minimum, you need to restore the data files that have been lost to the state they were in when they were last backed up.

3.1.1.5 Disaster

Many organizations have developed plans for recovery from disasters such as floods, fires, accidents, earthquakes, and terrorist attacks. You need to ensure that your strategy for backing up and recovering data fits in with any such plans. For example, you may need to arrange for backups to be made to a removable medium and stored offsite. The subject of disaster recovery is too broad for us to address in this book and is not discussed in any more detail.

The RDBMSs that we deal with in this book provide the facilities necessary to bring databases up to date by applying log files. They also provide the facilities necessary to undo changes made by partially completed transactions. This means that designers of database backup and recovery solutions do not need to concern themselves with recovering database data after statement failures or transaction failures. For each type of event that may occur, designers of a database backup and recovery solution must:

- Ensure that operational procedures specify who needs to do what to recover from loss or corruption of data used by the RDBMS.
- Ensure that the data files that the RDBMS recovery routines use are available when needed.
- Ensure that data that the RDBMS does not manage can be recovered to a state that is consistent with the database.

---

[1] We use the word “normally” here to highlight the fact that there are situations where further recovery actions may be required. One example of such a situation is when the failing SQL statement was issued by an application program that was performing a single business transaction. The application program wrote a record to a file system file that is not managed by the RDBMS. It then issues the SQL command, and this command fails. In this situation, the RDBMS will back out changes to tables managed by the RDBMS, but it cannot undo the changes made to the file system file. The responsibility for backing out changes made to objects that are not managed by the RDBMS must rest with the application program. We do not discuss this topic any further in this book.

3.1.2 Speed of Recovery

If you ask users how quickly they would like you to be able to recover lost data, they usually answer “immediately.” In practice, however, recovery takes time. The actual time taken depends on a number of factors, some of which are outside your control (for example, hardware may need to be repaired or replaced). Nevertheless, there are certain things that you can control and that will help to ensure that recovery time is acceptable:

- Develop a strategy that strikes the right balance between the cost of backup and the speed of recovery.
- Document the procedures necessary to recover from the loss of different groups or types of data files.
- Estimate the time required to execute these procedures (and do not forget the time involved in identifying the problem and the solution).
- Set user expectations realistically, for example, by publishing service levels that you are confident you can achieve.

3.1.3 Backup Windows

Some RDBMSs do not allow databases to be backed up while they are in use. In such cases, you need to shut down the database before the backup starts, and you cannot restart the database until after the backup has completed.

Shutting down a database often means that users cannot use applications. You need to ensure that the times at which databases are shut down and unavailable are acceptable to your users.

Even if you can perform backups while the database is operational, you need to ensure that any load on processors or networks caused by the backup process does not result in performance or responses that are unacceptable to your users.

3.1.4 Recovery Points

You need to define the points in time to which you will restore data. For example, you may need to recover the data to the state it was in when the last transaction was completed. Alternatively, it may be acceptable to restore the data to a consistent state that is no more than 24 hours old. In addition to either of these, you may be required to restore individual tables to the state they were in at any particular date within the last 30 days.

Whatever your situation, you need to consider recovery points and define a policy that is both achievable and acceptable to your user community.

3.1.5 Units of Recovery

In some circumstances, it may not be sufficient to restore individual tables (or even entire databases) to the state they were in at some point in the past. Sometimes, in order to maintain data consistency, you may need to restore data held in tables or files that have not been lost or damaged. This undamaged data needs to be restored to the same point in time as the damaged data.

In developing your backup strategy, you need to understand the relationships between the data objects on which user applications rely. For example, an order entry application holds information about orders in two tables. For fun, we build

this example using famous French impressionist painters as the customers and items in their paintings as the items they want to order. As shown in Figure 17 on page 35, the first table (ORDTAB) holds details such as the order number, customer number, customer name, and order date. The second table (ITEMTAB) holds the order number, product code, product description, and quantity ordered. If one of these tables is lost, it is essential that the other is restored to the same point in time. Otherwise, the database may hold inconsistent data:

- A row may exist in ORDTAB, indicating that an order exists, but there may be no corresponding details of items ordered in ITEMTAB. In Figure 17, Marc Chagall's order does not exist in ITEMTAB, as indicated by the pointing hand symbol. Perhaps his order for roof fiddlers has been lost, since he was famous for including them in his paintings!
- Several rows may exist in ITEMTAB, describing items that have been ordered, but with no corresponding row in ORDTAB to indicate which customer had placed the order. In Figure 17, someone has ordered Tahiti and someone has ordered dots, but there is no corresponding record in ORDTAB. You can tell this because order numbers 061 and 270 do not exist in ORDTAB. What famous painters might have placed an order for Tahiti and dots? (You guessed correctly if you said Gauguin and Serat!).

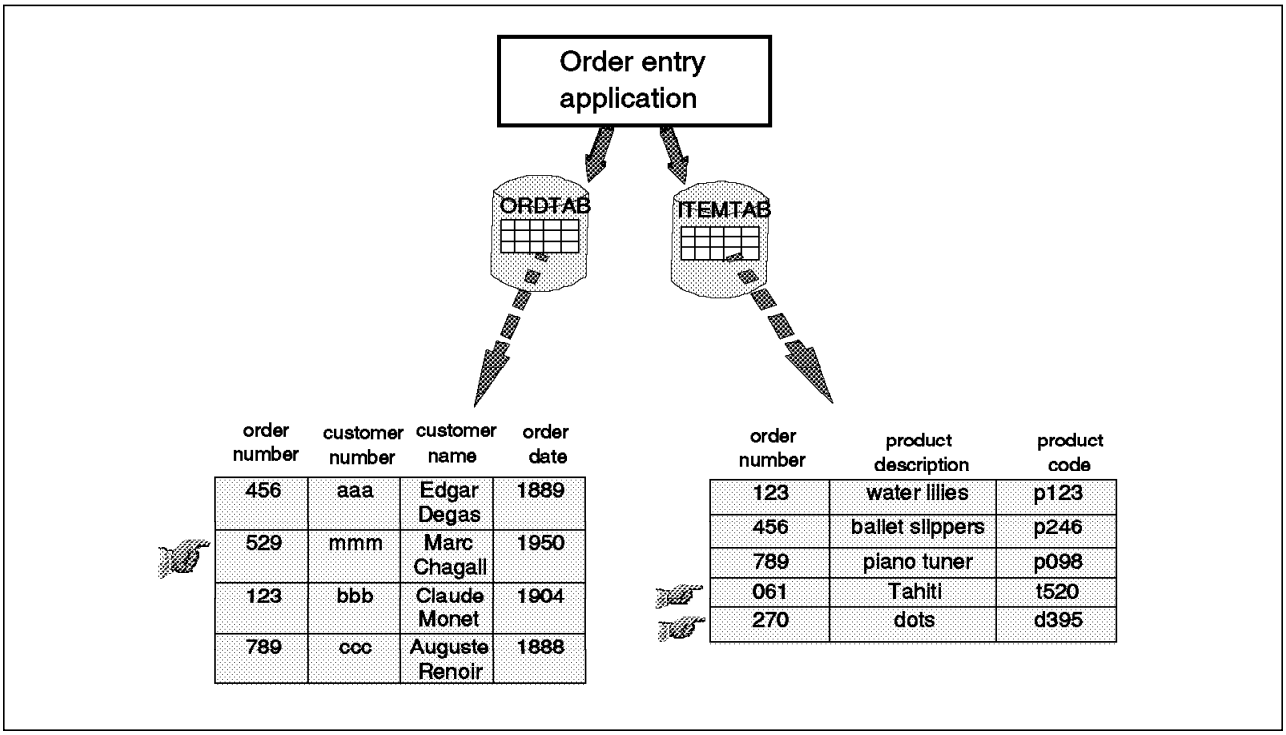


Figure 17. Order Entry Application Structure

Many applications rely on relationships that extend beyond the data held in a single database. For example, an engineering database application holds references to documents that exist as independent file system files. If the engineering database is lost and restored to the point in time at which the last backup was taken, references to documents may be lost. Alternatively, the medium on which some of the documents are stored may be damaged. If the data files used to hold the documents are restored to the point in time at which the last backup was taken, the engineering database may contain references to documents that do not exist. We look at this example in more detail in 3.5, "Advanced Implementations of RDBMSs" on page 43.

There are many other situations where you need to ensure that data consistency is preserved. The key point is that your backup and recovery strategy must take into account the needs of the applications that use the data.

3.2 Backup Techniques

You can use any number of techniques to back up data managed by RDBMSs. These techniques are, at least at a conceptual level, common to most RDBMSs. Often, a combination of techniques is used.

The purpose of this section is help you understand the techniques well enough to design an appropriate backup strategy. We examine how ADSM applies to these techniques and how each RDBMS product supports these techniques in Chapter 4, “Exploiting ADSM” on page 49 and in Chapters 5 through 9.

The techniques we consider are:

- Disk mirroring
- Offline backup
- Online backup
- Database export
- Full database backup
- Partial database backup
- Log file backup.

3.2.1 Disk Mirroring

Disk mirroring is a useful technique for maximizing the availability of your database. Mirroring is the process of writing the same data to multiple storage devices at the same time. This is done either sequentially, when data is only written to the mirror once the master write is successful or in parallel when both master and mirror writes occur at the same time. The first method is slower but you are more likely to have at least one good copy of the data if a failure occurs.

When reading from a mirrored logical volume, AIX will read from either the master or the mirror, whichever is the quickest at the time.

If a media failure occurs, operations are automatically switched to the good copy and AIX marks the faulty copy as stale. Mirroring allows your users to continue working even though a media failure has occurred. Mirroring can be implemented in either software or hardware; for example, AIX, most RDBMSs, and the IBM 3990 Model 3 storage controller provide mirroring facilities.

However, mirroring does not remove the need to back up databases. For example, disk mirroring will not allow you to restore a table that has been lost or damaged as a result of user error. Also, although disk mirroring dramatically reduces the impact of media failures, there is still a risk of damage to both sides of the mirror. If a database is held on one set of physical volumes, and a mirror image of the same database is maintained on a separate set of physical volumes, it is possible for both sets of physical volumes to be damaged or destroyed. This could happen as a result of a disaster or it could just be bad luck. In such instances, it will be necessary to recover the database from backup copies.

3.2.2 Offline Backup

Offline backup involves shutting the database down before you start the backup and restarting the database after backup is complete.

Offline backups are relatively simple to administer. However, they suffer from the obvious but significant disadvantage that neither users nor batch processes can access the database while the backup is taking place. You need to schedule sufficient time to perform the backup to ensure that the periods when the database will be unavailable are acceptable to your users.

Some RDBMSs provide a “single-user mode” or “quiesced mode.” You can think of this as an “almost offline” mode. A database administrator can still use the database, but general users cannot.

3.2.3 Online Backup

Some (but not all) RDBMSs allow backups to be performed while the database is started and in use.

Clearly, if a database is being backed up while users are updating it, it is likely that the data backed up will be inconsistent (“fuzzy”). The RDBMSs that support online backup use log files during the recovery process to recover the database to a fully consistent state. This approach requires that you retain the RDBMS log files and indicate to the RDBMS when you are about to start the backup and when you have completed the backup.

Some RDBMSs allow you to quiesce activity on portions of the database (for example, a particular table space) so that a set of complete tables is temporarily “frozen” in a consistent state. You then can back up the set of tables that has been “frozen.” Once the backup has completed, you can reactivate the table space.

3.2.4 Database Export

All RDBMSs provide export and import utilities. These utilities operate on logical objects as opposed to physical objects. For example, you can use an export command to copy an individual table to a file system file. At some later time, you might want to restore the table, in which case you would use the import command. Although export and import can be used for backup and restore operations, they are really designed for moving data, for example, for workload balancing or migration.

Note: You should assume that import will work only with files that have been created by the same RDBMS′s export utility.

Most other utilities operate on the physical data files that RDBMSs use to store their databases. Therefore, other utilities cannot normally be used to back up and restore a single table because:

- A single physical data file may contain data belonging to several tables.
- The data contained in a single table may be spread across multiple data files.

Thus, the only way to gain access to the set of data contained in a single table is through the RDBMS itself.

Export utilities are usually slower than most other utilities and should be used only when you need access to database objects or raw devices.

3.2.5 Full Database Backup

Full database backups involve making copies of:

- Data files used to hold user data
- Data files that hold tables used by the RDBMS itself
- RDBMS log files
- Any control files and parameter files that the RDBMS uses.

Many RDBMSs allow you to perform full database backup when the database is either online or offline. However, the technique for full database backup when the database is online can be quite different from offline.

To perform offline backup you can use the operating system utilities, RDBMS utilities, or ADSM to back up the data files that constitute the database. To perform online backup you need to use an RDBMS utility to create data files containing a copy of the database. You can then use ADSM to back up these data files together with the parameter files that you use to start up the RDBMS. We evaluate these approaches in more detail in Chapter 4, “Exploiting ADSM” on page 49and Chapters 5 through 9.

The simplest approach to database backup is to perform only full, offline backups at regular intervals. This approach is relatively easy to administer, and recovery is relatively straightforward. However, it may not be practical to take databases offline for the period of time that is necessary to perform full backups at the frequency you need. You may be have to adopt a more flexible approach.

3.2.6 Partial Database Backup

Many RDBMSs allow partial database backups when the database is online or offline.

Partial database backups involve backing up a subset of the full database (such as the data files that make up a table space). Make sure that the subset you back up (as part of a partial backup) represents a complete logical unit of recovery from the point of view of the application. Take the example described in 3.1.5, “Units of Recovery” on page 34; you should ensure that ORDTAB and ITEMTAB are backed up at the same time. You may also need to back up data files that the RDBMS does not manage.

You must also ensure that the unit of recovery is consistent from the point of view of the RDBMS. If you have added a new data file to a table space, you must ensure that any control file that the RDBMS uses to define the relationship between data files and table spaces is also backed up. This is discussed in more detail in 4.3.3, “Grouping Related Files” on page 59.

3.2.7 Log File Backup (Simulated Incremental)

For some applications, the units of recovery are too large to be backed up on a daily basis. Sometimes the constraining factor is the elapsed time that is available (the backup window). Sometimes the load that the backup would place on the network would have an unacceptably bad impact on other processes and users.

In such situations it may be possible to capture only the changes to the database by backing up the RDBMS′ log files. Some database vendors refer to log file backup as incremental backup, but it is really a “simulated” incremental backup as opposed to a “true” incremental backup. A true incremental backup backs up changed database blocks or pages, whereas a simulated incremental backup backs up the database transactions. Recovery from a simulated incremental can be much longer than from a true incremental because you must reapply all of the transactions in the logs.

To recover from a log file or simulated incremental backup:

1. Restore the database from a full database backup (in some circumstances, restoring from a partial backup may be sufficient).
2. Restore the log files.
3. Apply the log files to the restored database.

3.3 Additional Considerations

You may need to consider other factors for your database backup and recovery strategy, such as:

- Backing up to different types of media
- Breaking up units of recovery
- Restoring individual tables.

3.3.1 Backing Up to Different Types of Media

The media most commonly used for backup are:

- Diskette
- Tape
- Optical disk
- Magnetic disk.

The capacity of a diskette is too limited to be of practical use for backing up all but the most trivial database, so we do not regard diskettes as a serious option.

With regard to the other three media, technology is changing rapidly, and you should check with your supplier for details of specific products.

The speed of data transfer to magnetic disks makes them attractive for backup. However, most organizations find them too expensive for holding multiple copies of backed up data. The best way of using magnetic disk for backup is to use it as a temporary staging post; that is, you back up data initially to disk, and then, at some later point, a background process copies the backed up data to optical disk or tape. This approach has a number of benefits:

- Backup can be performed at magnetic-disk speeds (rather than slower optical-disk or magnetic-tape speeds).
- You do not have to wait for tapes or optical disks to be mounted before a backup can start.
- You do not have to incur the high costs involved in keeping all of your backed up data on magnetic disk.

In comparing optical disk with magnetic tape you should consider the following points:

- With suitable devices, you can now write many times to both types of media.
- Automatic loaders are available for both types of media.
- Some people argue that optical disk is more reliable than magnetic tape because it is less prone to stretching and snapping.
- The cost of storing data on optical disk is many times more expensive than storing the same data on tape.

3.3.2 Breaking Up Units of Recovery

Sometimes a unit of recovery may have such a wide scope that recovery of the entire unit is impractical or undesirable. In such cases, you may want to consider recovering only a subset of the unit of recovery. If you do this, you should have a program available that reads through the database and identifies any inconsistencies. It may be possible for the program to correct some of the inconsistencies. It should report all of the inconsistencies it finds and indicate whether it has taken any corrective action.

Some RDBMSs have a consistency-checking utility that provides some verification that the database is consistent. However, for many applications, you need to write your own programs to supplement such a utility. You may also need to develop special programs to manually rectify inconsistencies.

3.3.3 Restoring Individual Tables

You may be asked to restore individual tables (if, for example, they were accidentally dropped or corrupted) to the state they were in at some time in the past.

Before you consider restoring an individual table, you must check your definition of units of recovery to be sure that you are not in danger of introducing inconsistencies. You must also be sure that you are not creating problems for one group of users by solving the problems of another.

Having said this, you will normally be able to restore individual tables only if you had previously taken special action to back them up (or archive them).[2]

The only way to extract an individual table from a database for backup (or archive) is by using the RDBMS utilities themselves. Specifically, you will need

[2] The normal technique for backing up data is to back up the data files that comprise the RDBMS table spaces. If a table space consists of only a single table, the table could be restored simply by restoring the backed up files that constitute the table space. If, as is common, several tables share the same table space, then, by restoring the table space, you are in fact restoring all of the tables and not just the one with which you are concerned. We consider this issue in more detail in 4.4.2, “RDBMS Design Considerations” on page 63.

to use the RDBMS′ export command to create a file system file. You can then restore the table using the import command.

Of course, it is possible to use a variant of this technique, which involves the following steps:

1. Back up the table space.
2. Restore the table space from an earlier backup.
3. Export the table.
4. Restore the table space from the backup you have just taken.
5. Import the table that you have just exported.

However, this approach involves taking the table space, offline and in some situations this may not be possible.

In general, the preferred approach to restoring individual tables is to define a schedule for exporting specified tables at certain intervals. You and the users will have to agree on how long the exported copies should be retained.

3.4 Defining Your Backup Strategy

Here is an approach that you can use to develop a strategy for backing up and recovering databases:

1. Define units of recovery.
   a. List those objects (RDBMS tables and other files) that need to be recovered as a unit (unit of recovery).
   b. For each RDBMS table, list the name of the table space in which the table is held.
   c. List other tables that belong to the same table spaces and add the names of those tables to the unit of recovery.
   d. Identify other objects that need to be recovered to the same level as the tables that you have just added to the unit of recovery.
   e. Repeat steps 1a through 1d until you have identified all of the objects that make up the unit of recovery.
2. Define the relationships between tables, table spaces, data files, logical volumes, and physical volumes.
   a. List the names of the table spaces that contain the tables that make up the unit of recovery.
   b. List the data files that make up the table spaces.
   c. List the logical volumes on which the data files reside.
   d. List the physical volumes used by the logical volumes.
3. For each physical volume assess the impact of media failure.
   a. Which logical volumes need to be recovered?
   b. Which table spaces need to be recovered?
   c. Which other objects (data files) need to be recovered?
4. Assess whether your backup window is large enough to allow you to back up the unit of recovery.

a. Gather information about the initial size of the unit of recovery.

b. Gather information about how the unit of recovery is likely to grow over time.

c. Gather information about the volatility of the unit of recovery (how frequently does the data change?).

d. Decide how frequently you need to perform backups.

e. Produce a rough estimate of the time required to back up the unit of recovery.

   Of course, this time will depend on a number of factors, some of which you may not yet know. If you have some performance guidelines for estimating the time to perform backups, you should use them. In the absence of any such information, we recommend that you assume a backup rate of 0.5MB/sec.

f. Find out at what times the unit of recovery can be backed up and how much time is available to perform an offline backup of the entire unit of recovery.

g. Decide whether enough time is available to perform the backup.

   If there is enough time to perform the backup, apply the expected growth factors and calculate whether enough time will still be available. If there is still sufficient time, adopt a strategy of backing up the entire unit of recovery as often as necessary.

If frequent backup of the entire unit of recovery is not appropriate (because insufficient time is available or for some other reason), you can adopt one of three strategies:

- Perform online backup.
- Back up the entire unit of recovery at less frequent intervals (for example, weekly rather than daily) and back up only the RDBMS log files between those intervals.
- Break up the unit of recovery into smaller subunits.

These strategies are discussed in 3.2.3, “Online Backup” on page 37; 3.2.7, “Log File Backup (Simulated Incremental)” on page 39; and 3.3.2, “Breaking Up Units of Recovery” on page 40.

5. Estimate the time required to recover the unit of recovery and assess whether it is acceptable.

Repeat all of the steps outlined above for all other units of recovery, building up a composite schedule of the backups that you need to perform.

Now look for contention and conflict within the schedule. For example, if you intend to perform your backups to a single device, you need to consider the impact of performing multiple, simultaneous backups. You may decide to modify your strategy to accommodate such conflicts.

3.5 Advanced Implementations of RDBMSs

When you are designing a backup and recovery solution, it is easy to underestimate the true complexity of the problem and to devise a solution that fails to fully meet the real requirement.

The purpose of this section is to help you identify some of the hidden complexities of backing up and recovering databases and provide some hints on how to address them. We describe different ways in which databases have been implemented to meet business needs. We highlight aspects of the implementations that present particular challenges to the designer of a backup and recovery solution and suggest some techniques for dealing with these challenges.

3.5.1 Single Large Database

A utility company has a 40GB Oracle database implemented on a RISC System/6000. The database contains geographical data for the area covered by the utility company. The database needs to be available for online use between the hours of 6 a.m. and 10 p.m. from Monday to Friday. Overnight batch processing (excluding backup) takes approximately three hours. This leaves five hours each night for backup.

Assuming an effective data transfer rate of between 0.5 and 2.0MB/second,[3] it will take somewhere between 5.50 and 22.25 hours to back up the database. Both figures exceed the five-hour window available overnight for backups.

As you can see in Figure 18 on page 44, the solution that was adopted was to perform a full backup of the database on weekends and to perform an incremental backup each weeknight. Each night a shell script issued the ADSM *incremental* command to back up the log files that Oracle had created since the previous night. Given that no more than 20% of the database changed on any one day, and given the same assumptions about effective data transfer rate (0.5-2MB/second), the time required to perform an incremental backup was somewhere between 1.1 and 4.4 hours. Both of these figures are within the requirements.

[3] The effective data transfer rate is calculated by taking the total time taken to back up a file from disk to ADSM and dividing the total time by the size of the file as held on the user's disk. The figures quoted here are reasonable guidelines for a lightly loaded 16MB Token-Ring LAN connecting an adequately powered ADSM/6000 client to an ADSM/6000 server and carrying data compressed by the ADSM/6000 client. However, you should not apply these figures without first verifying them in your own environment.

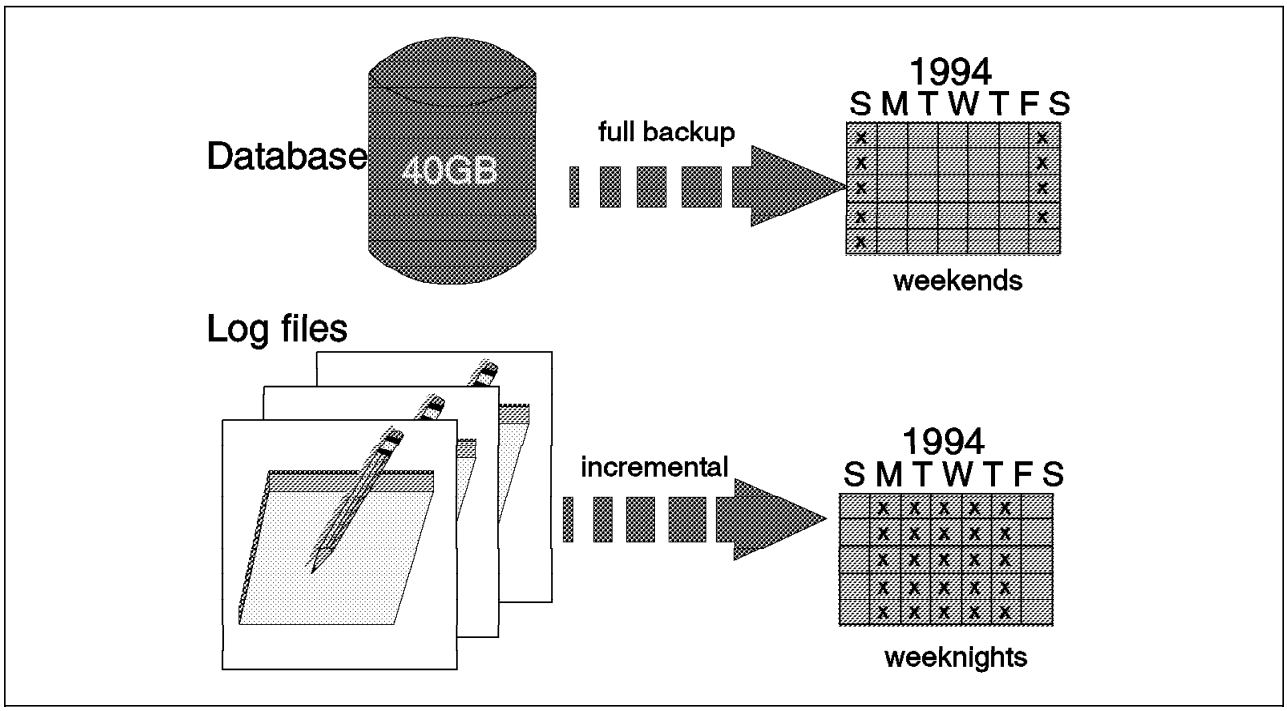


Figure 18. Single Large Database

3.5.2 Multiple Database Machines

A utility company developed a customer application to record details of consumption, billing, and payments. Several hundred people use this system to update and interrogate customer information. The company's customer service policy demands that this system be available 24 hours a day. The company implemented the application on a RISC System/6000 using DATABASE 2 AIX/6000.

At a later date the company decided to implement, on a separate Hewlett-Packard UNIX system, an image application to scan in and store customer correspondence. Users scan documents in any one of 20 scanners, and the application stores the images of the scanned documents on a database.

The disks that hold the image database are located in a building several miles away from the disks that hold the customer database.

The company regards the image application as less critical than the customer application. The image application needs to be available only between the hours of 7 a.m. and 7 p.m.

Having implemented both applications the company decided that, when clerks use the customer application to look at the details of a particular customer, the system should tell them whether there is any correspondence on the image database. The company achieved this by making the following changes:

- The image application updates the customer database whenever customer correspondence is scanned in.
- The customer database holds references to customer correspondence held by the image application.

- When users interrogate the customer database, the customer application indicates whether the image application holds any correspondence for that customer.
- If the image application holds correspondence for that customer and the user elects to view it, the customer application connects the user to the image application.

As shown in Figure 19, this implementation means that there are now logical links between two heterogeneous databases. These logical links are managed by application programs—not by either of the RDBMSs. This situation presents some challenges to the designer of backup systems.

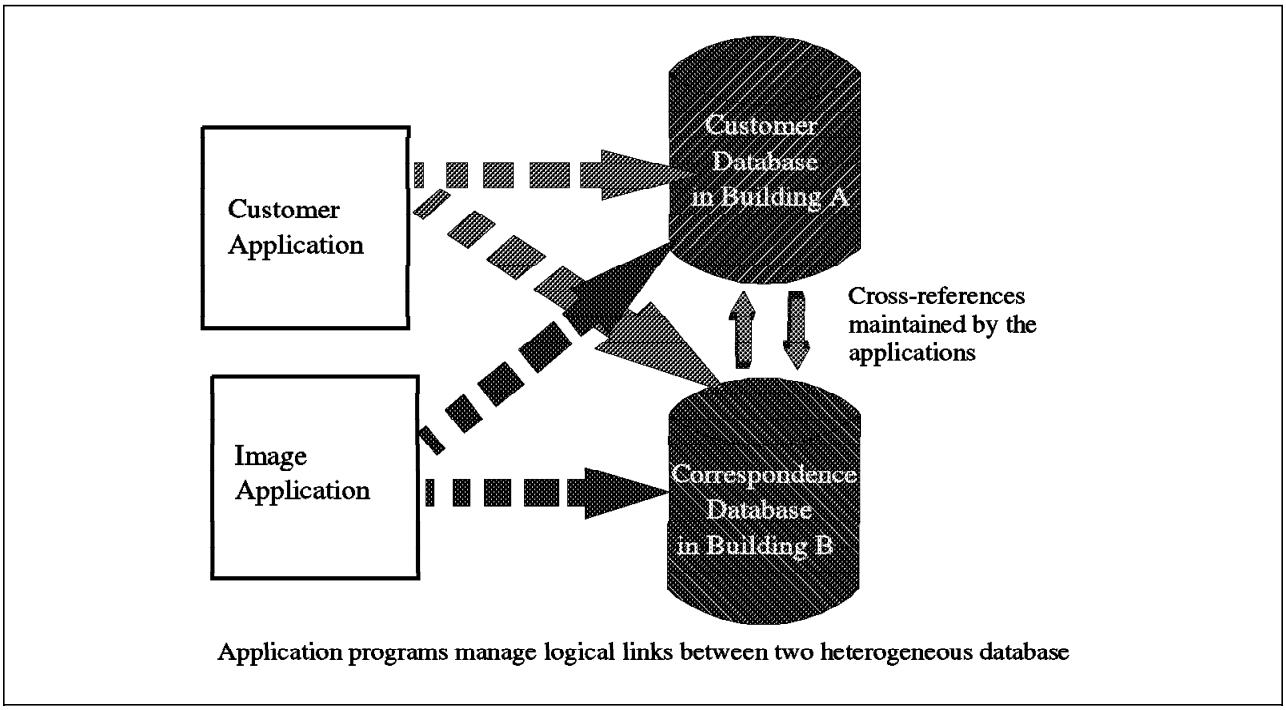


Figure 19. Multiple Database Machines

If a disaster destroys the image application's database as well as online log files, you will need to restore the image database from a backup copy. This latest backup copy may be up to one day old. If you restore the image database from a backup copy, the customer database will contain references to correspondence that no longer exists on the customer database. However, it is not practical to restore the customer database to the same point in time as the image database because:

- You will need to shut down the customer database, thereby making the customer application unavailable to users. This will infringe on the company policy of having the application available 24 hours a day.
- You will need to arrange for users of the customer database to reapply the changes they have made since the backup was taken.

Probably the best solution to this problem is to restore the image database using the latest backup and then run a specially written program to search through the customer database and check the references to items in the image database. When this program finds references to documents that no longer exist, it will update the customer database to remove those references. Users will need to

identify those documents that they had scanned in since the latest backup was taken and then use the image application to scan them in again. This approach maintains consistency between the databases and minimizes disruption to the users.

If a disaster destroys the customer database and it cannot be recovered, it will need to be restored from a backup copy. Once again there is a risk that the two databases will be inconsistent. In this case, the speed of recovery of the customer database will be paramount because the customer deems the customer application as more critical than the image application.

Our recommended approach when the customer database is lost and cannot be recovered is to restore the customer database from the latest backup and restore service to the users as quickly as possible. Users will then need to reapply the changes that they have made to the customer database since the backup was taken. Once users have reapplied the changes, you should run another specially written program that reads through the image database and checks that every document is referenced by the customer database. When the program finds a document in the image database that is not referenced by the customer database, it will create a reference by updating the customer database. This approach maintains consistency between the databases and minimizes disruption to the users.

3.5.3 Multiple Databases on the Same Machine

This situation is similar to that described in 3.5.2, “Multiple Database Machines” on page 44. The most significant difference is that in this case both the customer database and the image database are held on disks located in the same room in the same building. Thus, a single disaster could destroy both databases.

If both databases are destroyed, you would need to restore them from backup copies. You then need to find a way of getting the databases to a mutually consistent level. To achieve consistency between the databases, you should run two specially written programs. The first program removes references in the customer database to documents that do not exist in the image database. The second program searches the image database for documents that are not referenced by the customer database. When this program finds documents in the image database that are not referenced by the customer database, it creates references by updating the customer database.

Once you have put back the two databases into a consistent state, the users need to identify the data that has been lost and repeat the work required to bring the databases up to date.

The key point about this situation is that you can simplify your task of bringing the databases to a mutually consistent state and the users′ task of identifying the data that has been lost by synchronizing the backups of the databases. In other words, you should ensure that both databases are backed up at a time when they are mutually consistent.

3.5.4 Database with Links to File System Files

As shown in Figure 20, an engineering company has a database that contains pointers to documents produced during the design and manufacture of complex engineering constructions. These documents take many different forms. Some are on paper held in filing cabinets, some are items of machine-readable text and engineering drawings held as individual data files on magnetic disk, and still others are images stored on optical disk. The database may contain many references to the same document.

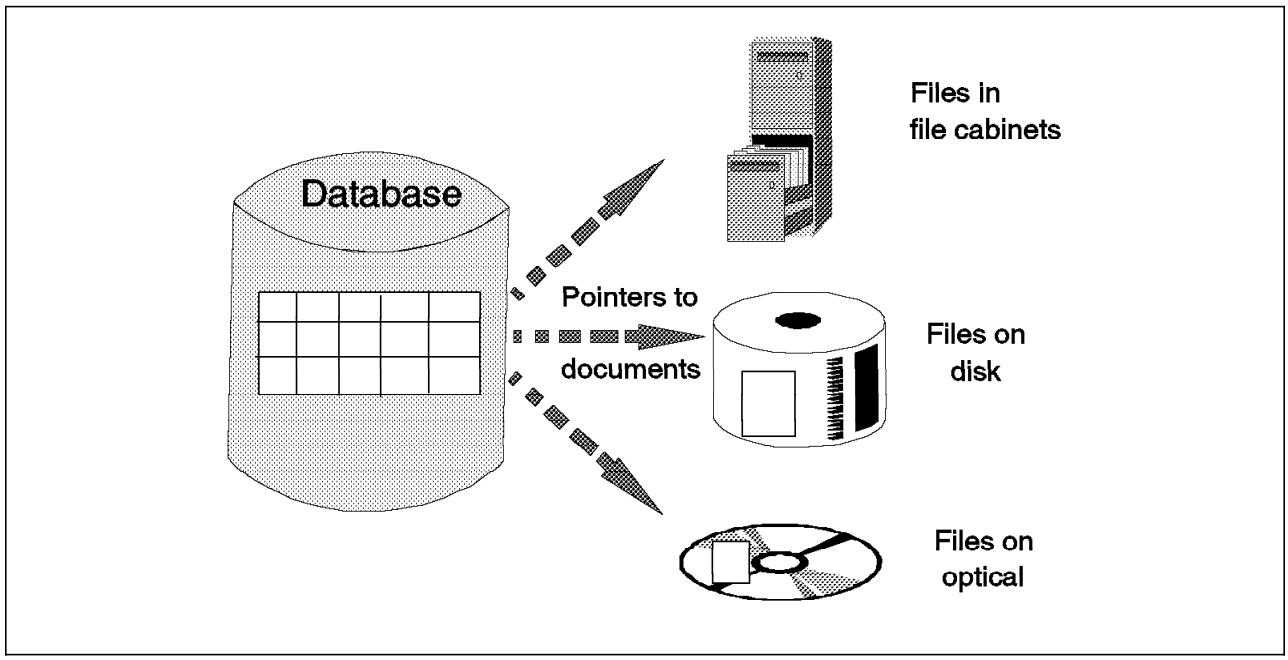


Figure 20. Database with Links to File System Files

If engineering drawings and text files are lost as a result of a head crash or other media failure, they can be restored from backup copies. However, the database may now contain references to documents that were created since the last backup was taken and do not now exist on disk.

To deal with this situation, we recommend that you run a specially written program that reads through the database looking for references to documents that do not exist. This program updates the database to remove these references. Users then need to re-create the documents that have been lost.

If the database has been lost and cannot be recovered, you need to restore it from a backup copy. It is not possible to identify what needs to be done to make the database consistent with the data files because the pointers that define the links between the database and the data files have been lost, and insufficient information is available to rebuild them either by program or by hand.

In this situation, probably the best thing you can do is restore the database and all data files to the same point in time. You then need to arrange for users to repeat the tasks that they have performed since the backup was taken.

We make two recommendations for this scenario. Our first recommendation is that you use disk mirroring to maintain two up-to-date images of the database and log files. Disk mirroring significantly reduces the likelihood of losing the engineering drawings and text files in the first place. Our second

recommendation is that you synchronize the backups of the database and the data files. If you ensure that the database and data files are in a mutually consistent state when you back them up, you simplify the users′ task of bringing them up to date.

3.5.5 Archiving Relational Databases

A utility company uses a customer database for several applications including its billing application. The database holds information about each customer including the customer′s name, address, and details of equipment installed. It also holds details for each bill presented to the customer and records of payments made by the customer.

Over time, as bills are produced and payments are made, the database grows. The utility company decides that it cannot justify the cost of keeping the details of all past bills and payments online for more than five quarters. However, the company does need to have access to billing and payment data for at least the past seven years. The company decides that it will keep, for each customer, details of bills presented and payments made within the previous 15 months (five quarters). When a new bill is prepared for a customer, details of any bill that is older than 15 months are archived to tape and deleted from online disk storage. Similarly, when a new payment is received from a customer, any payments that are older than 15 months are archived to tape and deleted from online disk storage.

All this seemed straightforward and simple until the application designer started to think about what users would need to do when they wanted to retrieve and query the archived data. The designer decided that users would want to use the same query facilities for archived data as they used for online, current data. They would also need the archived data to be structured in the same way as the online data. The designer also recognized that users would need access to customer information as it was at the time that the billing data was archived and not as it was at the time that the query was being performed. For example, a user might need to know what equipment was installed at the time the archived bill was produced.

The designer decided to address the problem by constructing a set of “archive” tables that matched the structure of the tables used for the operational data. After a user inserts a new bill or payment record into the operational tables, a batch program searches for bill and payment data that is older than 15 months. If it finds any old data of this type, it copies it, along with other relevant data, into the “archive” set of tables. It then deletes the original data from the operational tables.

When the batch program has finished, the “archive” tables contain a consistent snapshot of data at a particular point in time. The tables are archived to tape before being discarded. A new set of tables is created each time the batch job executes.

When users want to access archived data, they need to:

- Identify the set of data that needs to be retrieved
- Retrieve the data as tables
- Generate and run the queries against the tables.

Chapter 4. Exploiting ADSM

Now that you have an overview of ADSM, RDBMSs, and RDBMS backup techniques, you are ready to look at how ADSM can be integrated into your backup strategy.

First we position ADSM, RDBMS utilities, and the operating system utilities, in terms of general features and the types of data they support. Then we look at the various techniques you can use to back up databases with ADSM. Additional considerations, such as handling AIX sparse files, data compression, automated scheduling, why you might want to group files, and backup of the RDBMS software itself, are discussed.

We also discuss database design, including AIX design considerations involving the physical and logical volume layout as well as disk mirroring, and RDBMS design considerations, including the hierarchical structure we recommend.

We provide some examples of the use of the AIX commands **dd** and **tar** which can be used to back up some databases.

We close with a list of supported database products and the specific list of database products we worked with to develop this redbook.

4.1 Positioning of Backup Utilities

A variety of utilities can be used to back up databases. As we discuss in 4.2, “Techniques for Using ADSM to Back Up Databases” on page 51, many techniques for backing up a database entail using a combination of utilities. Therefore, let us position the utilities we discuss in this book. We look at ADSM, RDBMS utilities, AIX utilities (**cpio**, **tar**, **dd**, **backup**), and OS/2 utilities (**backup**, **xcopy**, and **pkzip2**). Table 1 provides an overall comparison of these utilities. Table 2 on page 50 compares these utilities on the basis of the types of data they support.

Feature	ADSM	User Programs with the ADSM API	RDBMS Utilities	AIX *cpio*	AIX *tar*	AIX *dd*	AIX *backup*	OS/2 *backup*	OS/2 *xcopy*	OS/2 *pkzip2*
Storage management	√	√								
Repository management	√	√								
Platform independence	√	√	√	√	√	√		√	√	√
Heterogeneous data	√	√		√	√	√	(√)	√	√	√

Table 1. Overall Positioning of Backup Utilities

The RDBMS utilities to which we refer in Table 1 are those basic utilities that are provided with the RDBMSs themselves. The specific utilities for each database product are discussed in the specific chapter on that database. We do not need to go into these details here to position the utilities with ADSM and the operating system utilities.

For a detailed explanation of the AIX **cpio**, **tar**, **dd**, and **backup** utilities and the OS/2 **backup**, **xcopy**, and **pkzip2** utilities, see Appendix B, "AIX and OS/2 Backup Utilities" on page 545.

Here are our definitions and explanations of the features we list in Table 1 on page 49:

- **Storage management** includes:
  - The ability to define storage hierarchies
  - Automated migration of data through the storage hierarchies
  - Automatic deletion of expired data
  - Automated space reclamation.

  Storage management is a key ADSM advantage not provided with RDBMS or AIX utilities.
- **Repository management** means that facilities are provided that allow the user to identify and locate backed up and archived copies of files.

  Repository management is another key ADSM advantage not provided with RDBMS or AIX utilities. Sometimes we refer to this feature as *system-assisted restore*. Users can restore their data using the system, without needing to pick up a phone and call a system administrator (live person!) for help.
- **Platform independence** means that the same utility will work on multiple platforms. Versions of the *cpio*, *tar*, and *dd* utilities are available on other platforms—but only on other UNIX platforms. ADSM goes further than the other utilities in this respect because it is available on many non-UNIX platforms.
- **Heterogeneous data** means being able to back up many different types of data. ADSM natively, or through use of its API, can back up any type of data it is sent. RDBMS utilities, however, are capable of processing only the data that is managed by that particular RDBMS.

Type of Data Supported	ADSM	User Programs with the ADSM API	RDBMS Utilities	AIX *cpio*	AIX *tar*	AIX *dd*	AIX *backup*	OS/2 *backup*	OS/2 *xcopy*	OS/2 *pkzip2*
RDBMS objects		√	√							
Data files	√	√	√	√	√	√	√	√	√	√
Raw devices		√	√			√		n/a	n/a	n/a
Sparse files	√	√	√	√	√	√		n/a	n/a	n/a

Table 2. Backup Utilities by Type of Data Supported

Here are our definitions and explanations of the types of data supported that we list in Table 2:

- **RDBMS objects** include tables, views, schemas, and indexes that are used by application programs and users. Table 2 indicates that neither ADSM nor the operating system utilities can back up RDBMS objects directly. However, there may be one type of RDBMS object that ADSM and the operating system utilities can back up—the table space. Table spaces consist of a number of data files. Sometimes one data file makes up a table space, and sometimes several data files make up a table space. Because all RDBMS data is held in table spaces and data files are never shared by more than

one table space, ADSM and the AIX utilities can back them up. An exception for ADSM is when the data is on raw devices. (We look at this case in more detail in 4.2, “Techniques for Using ADSM to Back Up Databases” on page 51.)

Neither ADSM nor the operating system utilities can directly back up individual objects that are held within table spaces. However, you can write programs that link RDBMSs to ADSM. These programs use SQL to communicate with the RDBMSs and the ADSM API to communicate with ADSM. Using these programs, you can retrieve objects from the RDBMS and back them up to ADSM.

One such program is *adsmpipe* which acts as an AIX filter and will take any data supplied to it and send it directly to ADSM. See Appendix C, “ADSMPIPE: Raw Logical Volume Backups with the API” on page 553 for more information.

- **Data files** are JFS files. All of the utilities can back up databases installed on JFS files. The RDBMS utilities operate on data at a logical level. The physical technique used by the operating system to store the data, such as JFS or raw devices, is irrelevant to the RDBMS utility. RDBMS utilities are effective with either data files, raw devices, or sparse files.
- **Raw devices** can only be backed up directly using the ADSM API, RDBMS utilities, or the AIX dd command. Applications that use the ADSM API shield the underlying data structure from ADSM. ADSM cannot directly access data stored on raw devices. However, if you first use either an RDBMS utility or the AIX dd command to create a JFS file from the raw device, ADSM can then back up that JFS file. This is an indirect way for ADSM to support raw devices if the application is not using the ADSM API.

  As mentioned above, RDBMS utilities do not care about the physical technique that AIX uses to store the data, so they can be used to back up data stored on raw devices. The AIX dd command can be used to copy the raw device into a JFS file.
- **Sparse files** are files in AIX that are defined as a certain size without actually reserving the disk space. ADSM (used with the proper options), most RDBMS utilities, and most AIX utilities can properly back up and restore sparse files. One exception is the AIX *backup* command. We describe and discuss backing up sparse files in more detail in 4.3.1, “Handling Sparse Files” on page 56.

4.2 Techniques for Using ADSM to Back Up Databases

Several techniques can be used to back up databases with ADSM. As you see in Figure 21 on page 52 and Figure 23 on page 56, the techniques can involve using the operating system utilities, the RDBMS utilities, and ADSM combined. Let us look first at using ADSM to back up databases installed on UNIX platforms (Figure 21 on page 52) and then at databases installed on OS/2 platforms (Figure 23 on page 56).

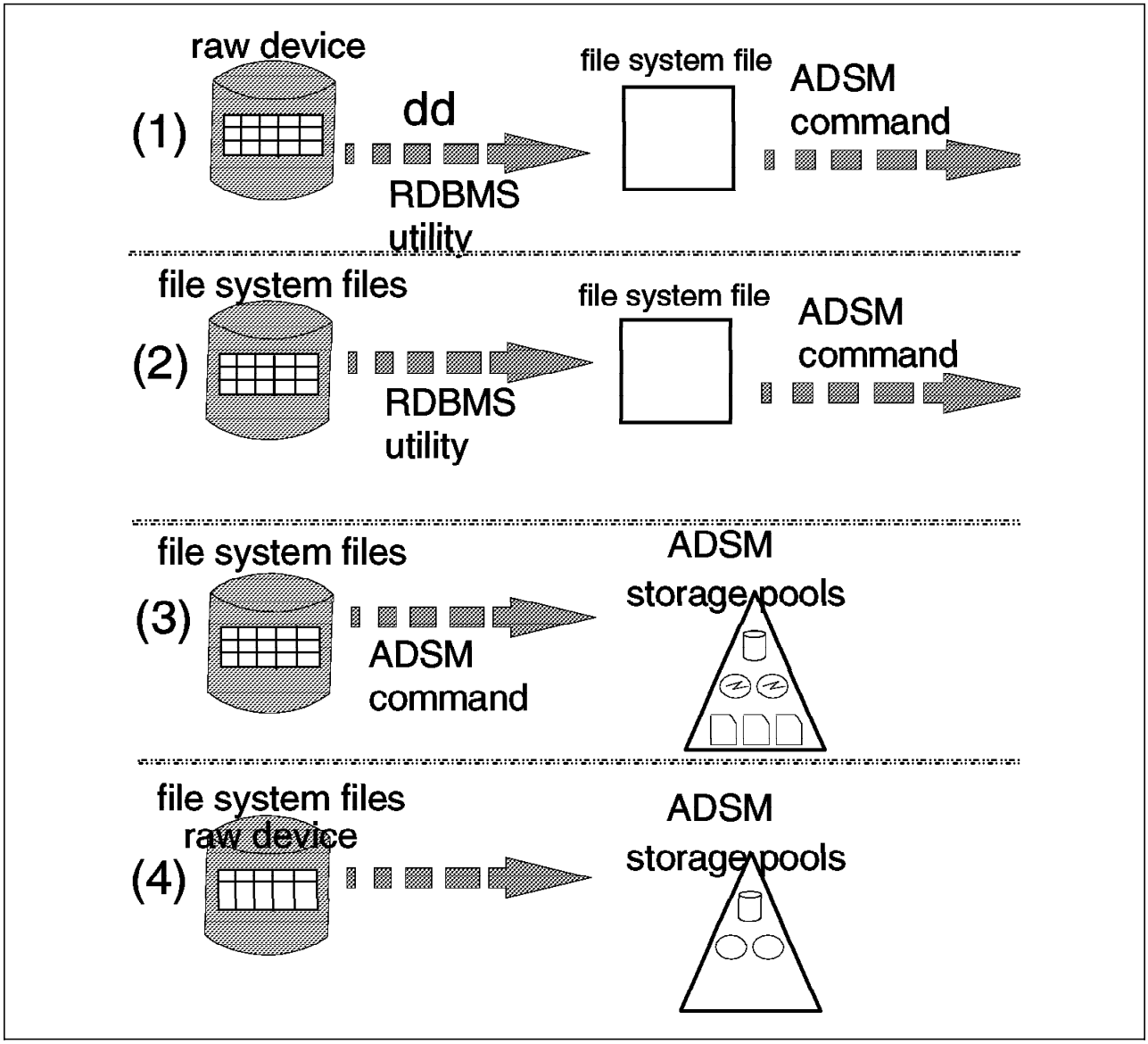


Figure 21. Techniques for Using ADSM to Back Up Databases on UNIX

In Figure 21, **technique (1)** applies to databases installed on raw devices. As mentioned in 4.1, “Positioning of Backup Utilities” on page 49, ADSM cannot read raw devices directly. If the database or application has not been enhanced to use the ADSM API, then, in order to use ADSM to back up a database installed on raw devices, you must convert the raw devices to one or more file system files. You can use the UNIX *dd* command for this conversion, but this command will back up the entire contents of the raw device, and your database must be offline. You can use RDBMS utilities to convert the raw devices, which will allow back up at the RDBMS object level and, depending on the database product, might allow online backup.

In either case, you need to be sure that you have sufficient disk space available to hold the file system files that the *dd* command or RDBMS utilities create. If the RDBMS utilities provide back up at an object level, such as table space, you might be able to reduce the amount of additional disk space you would need for the file system files. Remember also that, when you restore the data that has

been backed up by ADSM, you will need sufficient disk space to hold the file system files.

For more information about using AIX *dd* see 4.5, “Backing up Databases using dd and tar” on page 64.

Another consideration is that the current version of AIX does not allow a JFS file to be larger than 2GB. This may be an issue if you use an RDBMS utility to extract a database object that is larger than 2GB and try to write it to a JFS file. This may also be an issue if you use *dd* to copy a raw device that is larger than 2GB to a JFS file. This is not an issue for many RDBMS products, such as INFORMIX-OnLine Oracle, and Sybase, because they do not allow raw devices larger than 2GB, even though this is not an AIX restriction.

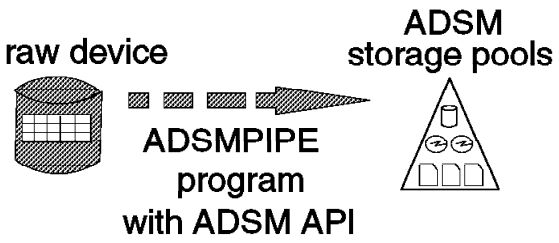
Technique (2) applies to databases installed on file system files. You may choose to use an RDBMS utility to back up the data to file system files and then use ADSM to back up those created file system files. You may choose this technique if you want to back up something smaller than the entire database, such as a table space, or if you want to take advantage of an RDBMS utility that allows online backups. The same considerations mentioned for **Technique (1)**, that is, enough disk space and the 2GB JFS file size limit, still apply.

Technique (3) also applies to databases installed on file system files. With this technique, you use ADSM to directly back up the file system files that make up the database and logs. The advantage to this approach is that no intermediate file system files are created, but the database must be offline to ensure a consistent copy.

Technique (4) applies to when the database has been installed on either raw devices or file system files. If the application or database product has been enhanced to use the ADSM API, the underlying physical structure is handled by the application, so it is not relevant whether raw devices or file system files are used. Also, the type of backup (for example, online, offline, at the table level) is determined and controlled by the application.

Another utility, ADSMPIPE, is provided as an “as-is” diskette with this redbook. ADSMPIPE aids in the backup of databases installed on raw devices. As shown in Figure 22 on page 54, ADSMPIPE reads raw devices and sends the data to ADSM using the ADSM API. It is an offline, full backup alternative. It is similar to using the dd command except that an intermediate file is not created. ADSMPIPE should be used *only* if no other better API alternative is available. For more information about ADSMPIPE, see Appendix C, “ADSMPIPE: Raw Logical Volume Backups with the API” on page 553.

ADSMPIPE for raw device backup



- As-is, unsupported tool with ADSM redbook
  - Solaris, AIX, and source code on
    - ★ diskette with redbook
    - ★ ftp.almaden.ibm.com anonymous ftp server
    - ★ IBM MKTTOOLS
- Offline backup alternative
- Most likely full backups only
- Use only if no other API alternative for your DBMS

PIPE

Figure 22. Techniques for Using ADSM to Back Up Databases on UNIX

Breaking a Mirror for Backups

Another technique for backing up databases on UNIX is to break one of the mirrored copies of a database and take a backup from that mirrored copy. The database is still online and available to users with the remaining mirror or mirrors. If you have three mirrored copies, you can still maintain a mirror for availability while you break the third mirror for backup. The technique is complex and prone to mismanagement, however, so we recommend using it only if you have no other alternatives.

Here are the steps for taking backups from a broken mirror:

1. Quiesce the file system before you break the mirror. If you do not quiesce the file system, the journal is not closed properly, and consequently the file system integrity cannot be guaranteed.
2. Take the mirrored copy offline and take a map of the logical volume partition layout so that you can re-create it when you resynchronize the mirrors.
3. Flush the disk cache before breaking the mirror; otherwise the cache data and disk data will not agree. The split mirror would be missing data and the backup would be of virtually no value.

The specific AIX steps would be:

1. unmount the file system
2. rmlvcopy the mirror (you can now remount the active file system)
3. reduce the vg (that is, take the disk with the broken mirror offline
4. make a new vg with this disk using the mapfile
5. varyon this vg
6. mount the file systems to a new mountpoint that you want to back up
7. perform the backup
8. unmount the mirror file systems
9. varyoff the vg
10. extend the original vg to include the disk again
11. mklvcopy the mirror back in again
12. syncvg

As shown in Figure 23 on page 56, similar techniques are used when ADSM backs up databases installed on OS/2 platforms. However, the techniques are a bit simpler because there is no concept of raw devices versus file system files. The databases can only be installed as part of the file system. There is also no file size limit of 2GB.

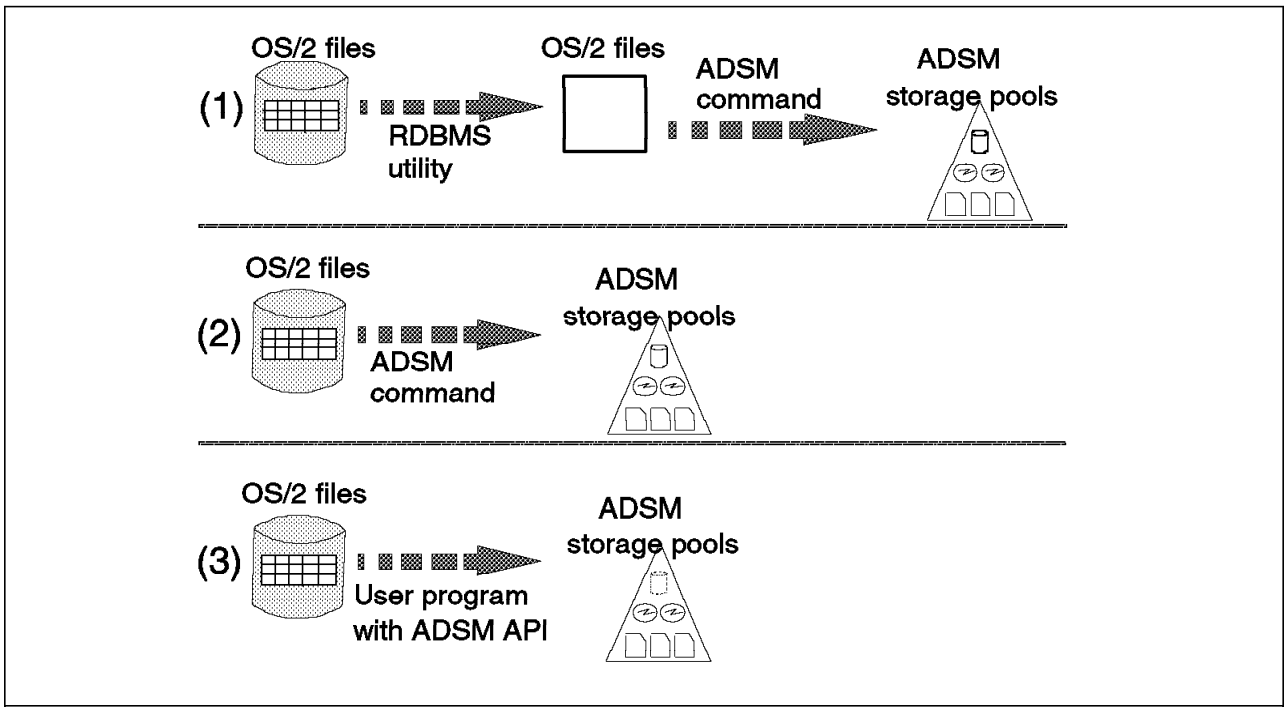


Figure 23. Techniques for Using ADSM to Back Up Databases on OS/2

4.3 Additional Considerations

There are other things to consider when defining your back up strategy and techniques. We look at how to handle sparse files, the data compression options available to you, how to implement automated scheduling of your backups, how to group several related files, how to handle the back up of the RDBMS executable files (the database software itself!), how you should set the ADSM copy serialization parameters, and how you can use the ADSM hierarchical storage manager (HSM) for the database log files.

4.3.1 Handling Sparse Files

The AIX JFS provides a feature known as *dynamic block allocation*. This feature allows you to define a file of a certain size without actually reserving the disk space. Disk space is acquired by the JFS as and when it is needed. For example, when an application wants to write data to a file, the JFS acquires the blocks on the disk that it needs for that data. Files of this type are called *sparse files*.

The benefit of sparse files is that disk space is used economically. For example, you may want to allocate 10 files. Each file may grow ultimately to 50MB. Initially, however, each file contains no more than 2MB of data. Without dynamic block allocation, you would define all 10 files as 50MB, and 500MB of disk space would be reserved for the use of these files even though initially only 20MB are used. Using dynamic block allocation, the files will use only 20MB of disk space. The remaining 480MB will be available for use either by these files as they grow, or by other files.

You should note that the AIX ls and df commands report the amount of disk space differently. The ls command shows the defined size of a file, whereas the

df command shows only the used parts of a file. Applying this to our previous example, the ls command would show a size of 50MB for each file, but the df command would show a size of no more than 2MB for each file.

When you add JFS disk space to most AIX-based RDBMSs, the RDBMSs initialize the space by filling it with binary zeros. When you back up these files, some utilities recognize that much of the file is filled with binary zeros and treat these blocks as though they had not yet been allocated. In other words, the binary zeros are not backed up. If, at some later stage, you restore this backed up data, the binary zeros will not be regenerated.

For some RDBMSs on some platforms, the loss of binary zeros may result in RDBMS internal errors. Other RDBMSs can continue to function perfectly well even though the binary zeros have not been restored. However, the loss of binary zeros can cause the database to run out of disk space unexpectedly. This situation can occur in the following way:

1. The database administrator allocates sufficient disk space for the database, say, 500MB.
2. The database is backed up using a utility that skips over binary zeros.
3. The database is restored from the backup copy. This effectively frees up the space formerly occupied by the binary zeros.
4. Other applications write data files and use the space freed up by the restore operation.
5. The database tries to acquire additional space to add new data to a table but finds all 500MB have been used up.
6. The database administrator is surprised and embarrassed because it appears as if all 500MB have been used up by the database, when in reality other applications may have used the space formerly occupied by the binary zeros.

The AIX for RISC System/6000 utilities that work this way are *backup* and *restore*. The AIX for RISC System/6000 *tar*, *cpio*, and *dd* utilities work differently and preserve the binary zeros.

The RDBMS utilities also preserve the binary zeros so they handle back up of sparse files correctly.

An undocumented ADSM Version 1 client option, MAKESParsefile, preserves binary zeros, so sparse files can be backed up correctly. You can supply this option as a parameter of ADSM′s restore or retrieve command, or you can include it within an options file. To preserve binary zeros when you restore or retrieve a file backed up by ADSM, set the MAKESParsefile option to “no.” For example:

```
dsmc restore -makesp=no "/oracle/dbs/system.dbf"
```

To set this option in the client options file, add the line: MAKESParsfile no. The default setting for the MAKESParsefile option is “yes.”

Please note that this option is required only if you use ADSM to directly back up the file system files that comprise the database and if you use an ADSM Version 1 client. If you use one of the techniques described in 4.2, “Techniques for Using ADSM to Back Up Databases” on page 51 where you use an RDBMS utility first and then use ADSM to back up the resultant file system file, the MAKESParsefile

option is not required. ADSM Version 2 clients are updated to automatically determine whether a file you are restoring or retrieving is a sparse file. The MAKESParsefile option is *not* required for ADSM Version 2 clients unless you are restoring or retrieving files that were backed up or archived with a Version 1 client. If you use a Version 2 client to restore or retrieve files that were backed up or archived with a Version 1 client, you must use the MAKESParsefile option.

Note: The MAKESParsefile option is not documented in any of the product manuals supplied with current releases of ADSM.

4.3.2 Data Compression

As part of a backup strategy, data compression is often used to save storage space, reduce network traffic, and sometimes even improve the performance of a backup and restore event. Data compression is especially important for backing up databases because you may have very large files to back up and restore.

There are many forms of data compression. Data can be compressed by the application (such as ADSM or SQL-BackTrack), the operating system (such as with the compress and pack commands in AIX), and/or specific hardware devices (such as the IDRC function of 3490Es).

You can use compression on the client using ADSM′s data compression facility, which gives approximately a 50% compression ratio. Because compression is done on the client platform, you need to consider the processing impact on the client, which depends on the power of the client. Typically, a fast processor on a slow line benefits from compression, and a slow processor on a fast line does not. However, if your key objective is to minimize storage space, you may not care about the size of the processor or the network speed.

You can use other compression facilities instead of ADSM, but is not recommended that you use both ADSM and other application compression utilities because compressing an already compressed file can cause the file to grow.

Let us look in more detail at how to use the ADSM data compression feature and the compression feature on a tape drive.

4.3.2.1 Using the ADSM Data Compression Feature

As mentioned above, ADSM′s data compression is done on the client platform. However, it is up to the ADSM administrator to decide whether data compression is always used, never used, or determined by the client.

These compression policies are defined for each client so you can decide, for example, to always compress files from your large database server and never compress files from your small end-user workstation. ADSM performs the compression for each file in real memory, so no additional data space is needed on the ADSM client.

The administrator can specify the COMPression option with the ADSM REGISTER NODE command. If the administrator decides that compression is client determined for that client (COMPression=Client), the client must also specify a COMPression option for the client′s options file. The compression option in the client′s options file is simply ON or OFF.

4.3.2.2 Using Data Compression for Tape Devices

If you have a tape drive installed on your machine that provides a compression feature, you could use this compression feature to store more information on a single tape, and, in many cases, you may get improved performance.

To use a compression feature on your tape drive, you need to specify compression on the ADSM DEFINE DEVCLASS command. For example, to use compression on an IBM 7208-011 8 mm tape drive, specify format=8500C on the *define devclass* command:

```
DEFine DEVclass tape DEVType=8mm FORMAT=8500C LIBRary=tapelib
```

To use compression on an IBM 3490E tape drive, specify format=3490C on the *define devclass* command:

```
DEFine DEVclass tape DEVType=CART FORMAT=3490C LIBRary=tapelib
```

4.3.3 Grouping Related Files

When backing up databases you may end up with groups of related files that you want to back up as a single unit. Grouping the files eases recovery. You may want to group all of the files that make up the database, or even if you have only a single database file, you may want to group it with log files or control files. Here are two approaches to group these files together.

One approach is to use an operating system command to package the files together and then use ADSM to back up that packaged file. For example, with OS/2 you can use the *pkzip2* utility to produce only one file containing all database objects. With AIX you can use the *tar* or *backup* utilities to package files together. The grouping of JFS files into a single package results in an *image file*. Image files are a very convenient way of handling large numbers of files as though they were a single unit.

Another approach is to use ADSM′s archive function instead of backup. Archive, as opposed to selective or incremental backup, can ease the task of managing all of the files that make the database a complete and consistent set. For example, you can use the *description* field or option to help identify the files that need to be retrieved as a group.

4.3.4 Backup of RDBMS Executable Code and Configuration Files

This book focuses mainly on the issues surrounding the backup of **data** that is managed by RDBMSs. We assume that you already have policies and techniques to restore software and configuration files in the event of their loss or corruption.

However, we do recommend that you use ADSM to back up the RDBMS executable code and configuration files. Keep in mind that each RDBMS has its own way of holding and using configuration data. For example, Oracle holds some configuration data in configuration files, some in parameter files, and some in control files. You should back up these files when they are first created and whenever they change thereafter.

As soon as you install and test a new release of the RDBMS software, you should use ADSM′s *archive* command to store a snapshot of the software, configuration files, and system data. You may find this archived snapshot useful if, at a later time, you encounter difficulties with the database.

4.3.5 ADSM Copy Serialization

There is one feature of the ADSM Copy Groups that you need to bear in mind as you back up your data.

There are four possible settings for ADSM Copy Serialization, *Static, Shared Static, Dynamic and Shared Dynamic*. Of these, we are interested in Static and Dynamic but we will describe all 4.

- Static

  If you set the serialization mode to static then none of the files you are attempting to back up must be in use. If they are in use then ADSM will not back them up.

- Shared Static

  This option is the default. Shared Static means that ADSM will attempt to back up a file if it is in use. However, if that file is modified during the backup process then ADSM will stop and try again. If it continues to be modified during backup then ADSM will not back it up.

- Dynamic

  The first attempt to back up any file will be accepted regardless of any changes made to that file during the backup process.

- Shared Dynamic

  This is similar to Shared Static in that ADSM will retry the backup of any file if it is modified during the backup process. However, it will commit the last attempt to back up the file regardless of any changes made.

The serialization mode is important if you are performing online backups of databases. You must set the serialisation mode to Dynamic or Shared Dynamic in order to back up an online database directly with ADSM. If the file to be backed up is “in use” by the database, even though the database is quiesced and the file is not being updated, then your backup will fail if you set the serialisation mode to Static or Shared Static.

Use Static or Shared Static if you are performing automated offline backups because the files you are backing up will not be in use by any program. Thus, if, for some reason your database did not close down successfully and your file is still in use, your backup will fail and you can issue a message to an operator.

4.3.6 ADSM Hierarchical Storage Manager

ADSM Version 2 provides hierarchical storage management (HSM), also known as space management, which is a combination of AIX or Solaris client and ADSM V2 server routines that automatically ensure that sufficient free storage space is available on your local client file systems. HSM migrates rarely accessed files to ADSM server storage. The files you use most frequently remain on your local client file systems, and those you use less frequently become eligible for migration to ADSM server storage. You can also specify that files be migrated to ADSM server storage on the basis of their size.

HSM works by mounting a special file system over the top of the JFS file system that a user normally sees and records whether a file has been sent to ADSM or not. It is not recommended that you use HSM for managing database files although there may be situations where the function of HSM would be useful. You should not use HSM for database files because there is likely to be

processing overhead. You would not want to migrate some of your database files to ADSM and keep other files local, without specifically designing which files could be safely moved remotely.

Database log files, particularly when you perform online backups of your database, can get quite large, and you may have problems with physically managing the log disk space. You may even run out of disk space. HSM can be very helpful with managing the log disk space. You can set up HSM to automatically migrate and recall the log files to ensure that you always have enough disk space.

4.4 Database Design Considerations

In this section we look at AIX and RDBMS design considerations. The logical and physical volume layout in AIX systems can affect your RDBMS availability, and the overall hierarchical structure of your RDBMS (if installed on a file system) can affect the ease of restoring data.

4.4.1 AIX Design Considerations

The Logical Volume Manager (LVM) facility of AIX for RISC System/6000 provides a very convenient way of managing disk layout. The LVM allows you to implement logical volumes on physical disk volumes. Each physical volume may contain several logical volumes, and each logical volume may be span across multiple physical volumes. A volume group is a collection of related physical volumes. The LVM frees you from many of the constraints imposed by the characteristics of your physical devices.

Let us look at two AIX design considerations that can affect the availability of your RDBMS: physical volume layout and disk mirroring.

4.4.1.1 Physical Volume Layout

The layout of your DASD can have a significant impact on the availability of your RDBMS. Looking at Figure 24 on page 62, consider, for example, a physical volume that holds several logical volumes. If the physical volume fails, all of the logical volumes on that physical volume become inaccessible. This is true for logical volumes that contain raw devices as well as logical volumes that contain JFS files. Consider now a single logical volume that spans multiple physical volumes. If one of the physical volumes fails, none of the data on the logical volume may be accessible (it depends on which database product you are using as to whether the data will be accessible).

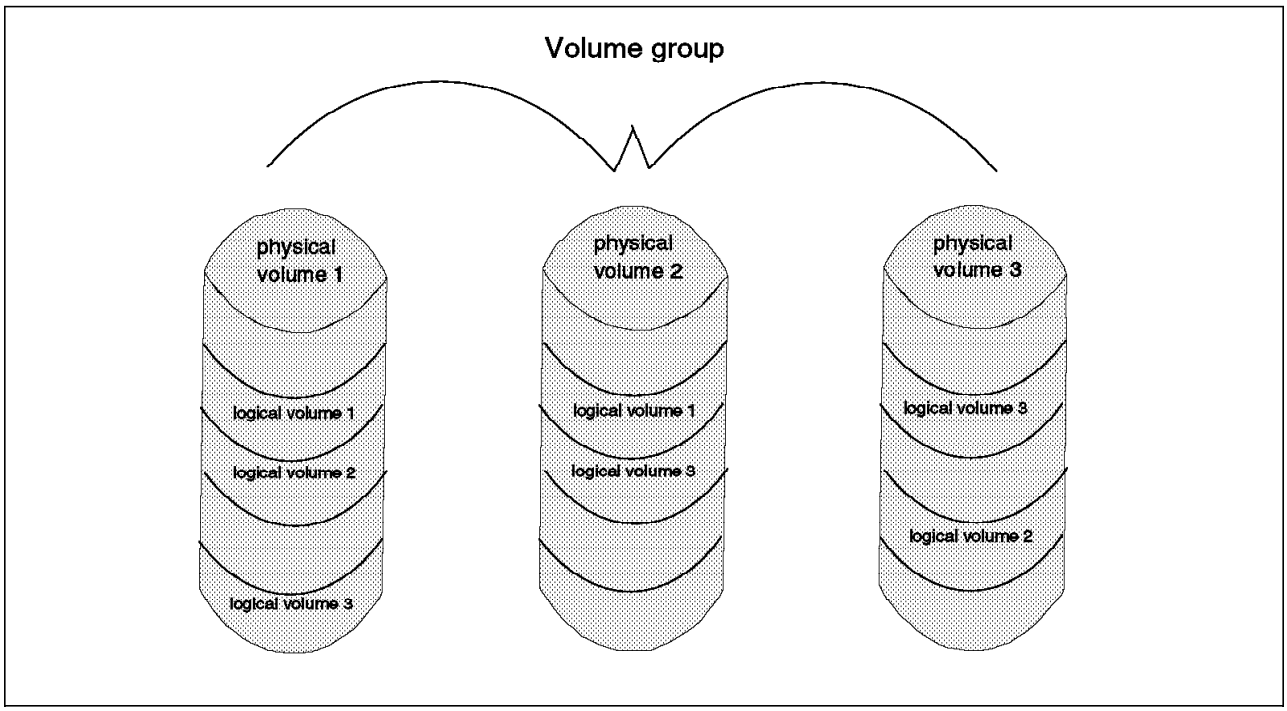


Figure 24. Logical Volume Manager

Imagine a scenario where all logical volumes span all physical volumes. In this scenario, a single disk failure would disable the entire system.

The key point is that by spanning logical volumes across multiple physical volumes you increase the risk of your data becoming unavailable. As a general rule, we suggest that you implement each logical volume on as few physical volumes as possible.

4.4.1.2 Disk Mirroring

Disk mirroring is a technique designed to maintain high availability. The technique involves maintaining two (or more) copies of data on disk. You can implement disk mirroring using the facilities provided by either the RDBMS or the LVM.

If the importance of your application justifies the cost of the additional disk space that mirroring requires, we recommend that you use LVM in preference to the RDBMS facilities concerning the performance. The performance of mirroring through the LVM facility is generally better than that provided through the RDBMS because on a read operation, the LVM will read the “fastest” copy if possible. The major advantange to mirror database files with the RDBMS facilities is that you can set up mirroring beyond the AIX volume group limits. The RDBMS facilities create, when implemented on AIX file systems, the second copy of database files in seperate AIX file systems, which you can define in different volume groups than the primary copy of the database files.

4.4.2 RDBMS Design Considerations

If you plan to install a database on JFS files, we recommend that you keep almost all of the files associated with the RDBMS in a hierarchy of directories under a **single parent directory**. Should the data be destroyed or corrupted, restoration of all files to a consistent state is easier if they can be addressed from a single directory.

You can spread the hierarchy of directories across several volumes by using multiple file systems, if you want. However, as shown in Figure 25, you should be able to address all files from a single parent directory. The hierarchy should include RDBMS executable code, configuration files, as well as data files. We recommend that the log files be held on a separate physical volume from the rest of the database files (see 4.4.2.3, “Log Files” on page 64).

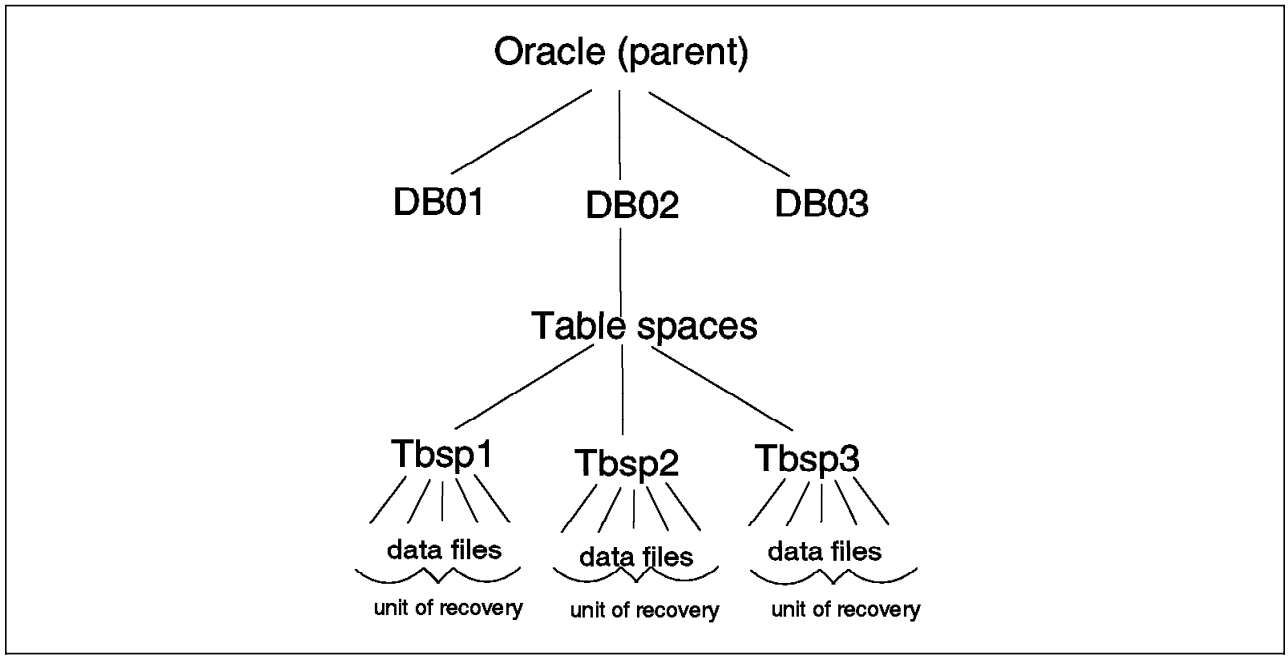


Figure 25. RDBMS Structure Recommendation

4.4.2.1 Tables and Table Spaces

Each RDBMS stores information about the data it manages in system tables. These system tables are just like any other relational database table. The RDBMSs create these tables in a table space we call the *system* table space. We recommend that the system table space be reserved exclusively for storing system data, not user data.

Most databases consist of multiple tables. Often there are interdependencies among tables such that if you restore multiple tables to the state they were in at different points in time, you will end up with an inconsistent database. As part of the process of developing a backup strategy, you should avoid the generation of such inconsistencies. One technique for achieving this is to define units of recovery.

We recommend that you maintain a one-to-one relationship between units of recovery and table spaces. In other words, as shown in Figure 25, we recommend that each unit of recovery be held in a single table space and that each table space represent a single unit of recovery.

4.4.2.2 Table Spaces and JFS Files

When you implement a database on JFS files, we recommend, as shown in Figure 25 on page 63, that you create a directory for each table space and keep each data file associated with that table space in that directory. Do not use the directory for anything other than data files for that particular table space. We also recommend that you define each table space in its own file system to capitalize on the one-to-one relationship between the logical volume and the file system. By doing this, the table space will also have a one-to-one relationship with the logical volume. However, if the table space is larger than 2GB, you may want to place each data file in its own file system. By doing this, the data file will also have a one-to-one relationship with the logical volume.

In the following example, the tspace01 directory contains three data files that hold the data for one Oracle table space, and the tspace02 directory contains two data files that make up a second Oracle table space:

```
/usr/oracle/orfs/tspaces/tspace01/datafl01.dbf
/usr/oracle/orfs/tspaces/tspace01/datafl02.dbf
/usr/oracle/orfs/tspaces/tspace01/datafl03.dbf

/usr/oracle/orfs/tspaces/tspace02/datafl01.dbf
/usr/oracle/orfs/tspaces/tspace02/datafl02.dbf
```

We recommend that you group data files used for table spaces in this way for two reasons. It is easier to locate the files that need to be restored when restoring the table space, and, if you adopt our recommendations, you can be sure that all files are on the same logical volume and a minimum number of physical volumes. As we explain in 4.4.1.1, “Physical Volume Layout” on page 61, this reduces the risk of losing availability of the table spaces. Moreover, if different table spaces are implemented on different physical volumes, the loss of one volume may result in only one or two table spaces becoming unavailable. The rest of the database may be able to continue operating normally.

Unfortunately, the table spaces we defined in our project environment (and that you will see in the examples in Chapters 5 through 12) do not conform to our recommendations made above. We developed these recommendations based on the experiences we had during the project.

4.4.2.3 Log Files

Log files are among the most sensitive and important data files for an RDBMS. We recommend that you use LVM to mirror the log files and that at least one copy of each log file be held on a separate physical volume from the table spaces.

4.5 Backing up Databases using dd and tar

It is possible to use the AIX dd and tar commands to back up raw logical volumes and journaled file systems to ADSM, as shown in Figure 21 on page 52. These commands can be used with any of the database products to produce an offline backup.

Use dd and tar with care. Ensure that the data you are backing up is sufficient to allow you to recover your database; for example, you should back up logical units of recovery. Therefore, for all database products, dd and tar are only useful for producing full, offline backups. Logical units of recovery are ensured

because the database is offline and its data storage areas are closed and consistent; all transactions are either completed or rolled back.

One further consideration: the use of dd and tar assumes that the data to be stored does not exceed the maximum AIX file size of 2GB. If the data exceeds this 2GB limit you must either divide the backup into more files or use another technique to back up your data.

Use dd or tar to back up your database only if a better alternative is not available. The best alternative is an RDBMS or third-party vendor backup utility that has been updated to use the ADSM API. With such an alternative you will most likely be able to do online backups directly to ADSM without having to create an intermediate file.

Note: You can also use the *ADSMPIPE* program for offline backup of databases installed on raw devices to ADSM. Use this alternative only if a better alternative is not available for your database product. For more information on this program see Appendix C, “ADSMPIPE: Raw Logical Volume Backups with the API” on page 553.

4.5.1 Full Offline Backup Using AIX tar and ADSM

You can use the AIX tar command and then ADSM to back up an INFORMIX-OnLine image file. When you back up using tar the database server must be offline to avoid possible inconsistencies between dbspaces during the backup. It is not necessary to back up any log files because log forward recovery is not an option when the data is recovered from an offline backup.

The AIX tar command handles sparse files correctly.

As shown in the following example, first we use tar to back up the INFORMIX-OnLine data to an image file called /home/informix5.image, and then we back up the image file with the ADSM dsmc selective command:

```
tar -cvf /home/informix5.image /usr/informix5
dsmc selective -password=mars -quiet /home/informix5.image
```

The following commands are the corresponding ADSM restore and AIX tar commands to restore the information:

```
dsmc restore -password=mars -quiet /home/informix5.image
tar -xvf /home/informix5.image /usr/informix5
```

4.5.2 Full Offline Backup Using AIX dd and ADSM

ADSM cannot directly back up a raw logical volume. One way of backing up databases installed on raw devices is to convert the contents of the raw devices into file system files, using the AIX dd command, and then use ADSM to back up the file system files. The dd command retains the internal structure on raw devices when you copy them to a file system file, but you do have to ensure that you have adequate disk space. For more information on the dd command, see Appendix B, “AIX and OS/2 Backup Utilities” on page 545.

As for a backup using tar, ensure that your database has been taken offline and that the logical volume is closed.

4.5.2.1 Backing up using dd and ADSM

Before you can use ADSM to back up a Sybase database that is installed on a raw device, you have to convert the data on each raw device to a file system file with the AIX *dd* command.

The AIX dd command creates sparse files when copying and converting the contents of raw devices to file system files. Therefore use the ADSM dsmc restore command with the MAKESParsefile=no option when restoring the files with ADSM. We also recommend that you be root user when using the AIX dd command.

We assume, in this example, that both the database data tables and transaction logs are on the same logical volume.

Before you can use dd you must first identify on which raw logical volume your data resides. Then use the dd command to create a file.

We use the following dd command:

```
dd if=/dev/rdb00 of=/home/sybase/db00.dd
```

Figure 26 shows the results.

```
dd: 32768+0  records in
dd: 32768+0  records out
```

Figure 26. Results of AIX dd Command

Make sure that the ADSM compression feature is enabled before backing up this file. The dd command performs a device-to-device copy, and the output file is the same size as the logical volume you are converting, even if no data is stored on it. We use the following ADSM command:

```
dsmc selective ″/home/sybase/db00.dd″ -password=mars
```

Figure 27 shows the results.

```
Selective Backup function invoked.

Session established with server BALTIC: AIX-RS/6000
 Server Version 1, Release 2, Level 0.0
 Server date/time: 06/16/1994 16:06:02  Last access: 06/16/1994 16:05:55

Normal File-->      16,777,216 /home/sybase/db00.dd . Sent
Selective Backup processing of ′/home/sybase/db00.dd′ finished with no
failures.
```

Figure 27. Results of ADSM Command to Back Up Output of AIX dd Command

Once your dd command has completed, the database can be restarted. The ADSM command to back up the file can take place in parallel. This is one way of reducing the amount of time needed to back up files.

4.5.2.2 Restoring Using ADSM and dd

To restore a database previously saved with ADSM and dd you must again ensure that your database server is shut down. Use ADSM to restore the database files and then dd to convert the files back to raw devices.

Use ADSM to restore the database files:

```
dsmc restore "/home/sybase/db00.dd" -password=mars
```

Figure 28 shows the results.

```
Restore function invoked.

Session established with server BALTIC: AIX-RS/6000
 Server Version 1, Release 2, Level 0.0
 Server date/time: 06/16/1994 16:13:01   Last access: 06/16/1994 16:11:1

File /home/sybase/db00.dd already exists, do you want to replace
it (Yes/No) y
Restoring   16,777,216 /home/sybase/db00.dd  . Done
Restore processing finished.
```

Figure 28. Results of ADSM Command to Restore the Database Files

Use the AIX *dd* command to copy back the file you obtain to the logical volume (raw device); in other words, convert the file into the raw device.

The following is the dd command we use:

```
dd if=/home/sybase/db00.dd of=/dev/rdb00
```

Figure 29 shows the results.

```
dd: 32768+0  records in
dd: 32768+0  records out
```

Figure 29. Results of AIX dd Command to Copy Back Database to Raw Devices

Finally, restart your SQL server.

Chapter 5. Automating Database Backups

This chapter describes some of the ways that you can automate your database backups. We look at **cron** and how you can use it to automate your backup. We look at OS/2 and how you can automate OS/2 backups. We also look at the ADSM central scheduling facility and how it can be used to automate the backup of a database. Some database backup products, such as SQL-BackTrack, provide their own automation facilities. Details on how to use their automation facilities are in the product documentation.

Detailed information about the ADSM central scheduling component is not included in this redbook. Detailed information is in Chapter 7, “Scheduling Operations” of the *ADSM Administrator’s Guide,* and Chapter 5, “Automating ADSM Tasks” of *ADSM: Using the UNIX Backup Archive Clients*. However, we do describe the functions and concepts of the central scheduler in this redbook.

The examples shown here include some simple shell scripts written to automate parts of the backup process. The scripts do not include any error checking or automation recovery as those processes depend heavily on the configuration of your databases and how you want to structure your backups.

5.1 Using cron to Automate Database Backups

Here we look at the use of the AIX **cron** utility to automate the backup of an INFORMIX-OnLine database. This example could apply to any database where you write a script to produce your backup and run ADSM.

5.1.1 Why Use cron?

Scheduling events for unattended backup activities is a common way of automating storage management procedures. Automating backup and archive operations ensures data consistency and safety for every client node.

The AIX **cron** utility provides some flexibility for automating operating system or RDBMS tasks such as:

- Ensuring the operating system is in a definitive mode, for example, in single-user, multitasking mode, so that no one can change data during the backup operation
- Starting dedicated processes or preliminary tasks, for example, mounting some AIX file systems that have to be stored or canceling applications that are hung
- Bringing the RDBMS to an offline or other mode, for example, quiescent, so that no one or only the database administrator has access to it.

Here we show the use of cron to issue ADSM selective, incremental, or archive commands.

5.1.1.1 What Is the AIX cron Utility?

The AIX cron utility runs shell commands at specified dates and times. It consists of two parts: the cron daemon and the crontab file.

The cron daemon checks every minute for an entry in the user-related crontab file (in the /var/spool/cron/crontabs directory) that can be executed. If the time specified in the entry matches the current time, the entry will be executed. For example, the example below shows an entry in a root user crontab file where a shell script called /etc/stop_db is performed every day at 01:30 (a.m.). Use the AIX command, *crontab -l*, to display the contents of the crontab file.

```
# crontab -l
30 1 * * * /etc/stop_db
```

Each crontab file entry consists of a line with six fields, separated by spaces and tabs. The fields contain, respectively:

1. The minute (0 through 59)
2. The hour (0 through 23
3. The day of the month (1 through 31)
4. The month of the year (1 through 12)
5. The day of the week (0 through 6 for Sunday through Saturday)
6. The shell command.

Cron starts a limited **bsh** (Bourne shell) for each command user. Verify that your shell script matches the *bsh* syntax.

5.1.1.2 Using cron with ADSM: Example

Figure 30 on page 71 shows a simple shell script that you can initiate using cron. The shell script processes an INFORMIX-OnLine database implemented on raw devices. The shell script:

- Shuts down the database
- Exports the data to a file
- Uses ADSM to back up the resultant file.

You will probably have to modify this shell script to:

- Remove the *-quiet* option
- Direct the ADSM report to a file
- Analyze the file to confirm that the backup (or export in our example) has worked successfully
- Restart the database.

```
# Shutdown the database server
su - informix "-c tbmode -ky"

if test $? != 0
   then
   echo "shutdown of the database failed"  >/dev/console
   exit 1
fi

# Unload the contents of the database
su - informix "-c dbexport -o /usr/informix5 inf5testdb01

if test $? != 0
   then
   echo "unload of the database failed"  >/dev/console
   exit 2
fi

# Activate the ADSM dsmc selective command to backup the Informix database
dsmc selective -quiet -password=mars -subdir=yes /usr/informix5

if test $? != 0
   then
   echo "backup of the ADSM client failed"  >/dev/console
   exit 3
fi

exit 0
```

Figure 30. Sample Shell Script Initiated by cron

The tbmode and dbexport database commands in the shell script are specific to the INFORMIX-OnLine Version 5.0 RDBMS. Refer to the particular RDBMS manuals or the database chapters in this book to determine the commands that you should implement in these scripts.

5.2 ADSM Central Scheduling Facility

The ADSM central scheduling facility provides a number of different ways of automatically scheduling data backups. Central scheduling lets you manage one or more administrative or client operation schedules. In this chapter we look at client operation schedules used to automatically back up databases.

5.2.1 Scheduling Modes

There are two possible ADSM scheduling modes: client-polling and server-prompted.

5.2.1.1 Client-polling

In client-polling mode, the client polls the server at regular intervals to determine whether there are scheduled operations to execute. Any ADSM client can use this mode.

5.2.1.2 Server-prompted

In server-prompted mode, the client waits for the server to prompt it to execute the next operation. Only clients communicating with the server through TCP/IP can use server-prompted mode.

A server can be set to scheduling mode **any**, which enables it to provide scheduling facilities to both server-prompted and client-polling ADSM clients. To set the server scheduling mode, either use the ADSM administrative GUI by clicking on the *Central Scheduler* icon or issue the *set schedmodes* command.

To set the client scheduling mode use the *schedmode* option in the client system options file, dsm.sys. You cannot change the client scheduling mode from the command line.

5.2.2 Scheduling Options

There are a number of options for tailoring the scheduler workload. They provide a means of ensuring the most efficient use of the ADSM server and, at the same time, ensuring that the data from multiple clients is backed up. Some of these options are server options, others are client options. The options are:

- Maxschedsessions

  The maxschedsessions option on the server defines the maximum number of concurrent client/server sessions. It specifies which percentage of the maximum concurrent sessions (maxsessions) can be used by the scheduler. For example, *set maxschedsessions 25* limits the scheduler to 25% of the total available client/server sessions. The default is 50%.

- Randomize

  The *randomize* option scatters the start times of schedules across the available startup window. The startup window is the start time and duration during which a schedule must be initiated. The randomize value is the percentage of the startup window for which you want the start times to be scattered. For example, if you set the randomize option to 50%, the schedule start times are scattered over the first half of your startup window. The default is 25%.

- Queryschedperiod

  The *queryschedperiod* option establishes how often client nodes contact the server to obtain a schedule. The default is 12 hours.

- Maxcmdretries

  The *maxcmdretries* option establishes the number of times that the client scheduler retries a failing schedule. The default is 2 retries.

- Retryperiod

  The *retryperiod* option establishes the number of minutes to elapse before the client scheduler retries a failed schedule. The default is 20 minutes.

We did not make any changes to these values during our testing. We recommend that you experiment with these values to tune your scheduling activities.

5.2.3 Scheduling Event Types

You can schedule seven different client actions. They can be set in the ADSM administrative GUI event properties notebook. They are:

- ARCHIVE

  Archives the files specified in the objects entry field.

- COMMAND

  Executes, on the client, the command specified in the objects entry field.

- INCREMENTAL

  Executes an incremental backup of files on the client.

- MACRO

  Executes an ADSM macro (a collection of ADSM client commands).

- RESTORE

  Restores the files described in the object entry field.

- RETRIEVE

  Retrieves the files described in the object entry field.

- SELECTIVE

  Backs up the files described in the object entry field.

5.2.4 Using the ADSM Central Scheduler

The simplest way of scheduling an automated backup of a database is to write a short script to perform the backup and then use the ADSM central scheduler to issue a switch user (su) command to execute the script.

Our example backs up the Sybase pubs2 sample database, but it could be easily changed to back up data from any other database manager.

The approach used is to define an ADSM schedule that issues an *su* command with a script file as a parameter. The script file pipes a list of Sybase SQL commands from a file into the Sybase *isql* program. Because the Sybase backup utilities do not send the data directly to ADSM through the ADSM API, the script then runs a dsmc selective command to back up the files created. If you use a backup utility that uses the ADSM API, you do not need the dsmc selective step. Figure 31 shows our sample script file, autoback, and Figure 32 on page 74 shows the SQL statements (which are in the dbbackup file).

```
# Autoback script file to create a backup of pubs2 database
# First remove the old backup files
rm /u/sybase/pubs2.*

# Now run the backup command
isql -Usa -Ppassword -Ssybase < /u/sybase/scripts/dbbackup

# And back up the files using ADSM
dsmc selective "/u/sybase/pubs2.*" -password=mars
```

Figure 31. Sample Autoback Shell Script

```
dump database pubs2 to "/u/sybase/pubs2.dump"
go
dump transaction pubs2 to "/u/sybase/pubs2.tr01"
go
```

Figure 32. Sample SQL Statements in dbbackup File

To define the ADSM schedule, use the ADSM administrative GUI; select Backup/Archive Schedules and then select Edit from the action bar. As shown in Figure 33, select Add from the pull-down window, which opens a notebook to add a new scheduler event (not shown).

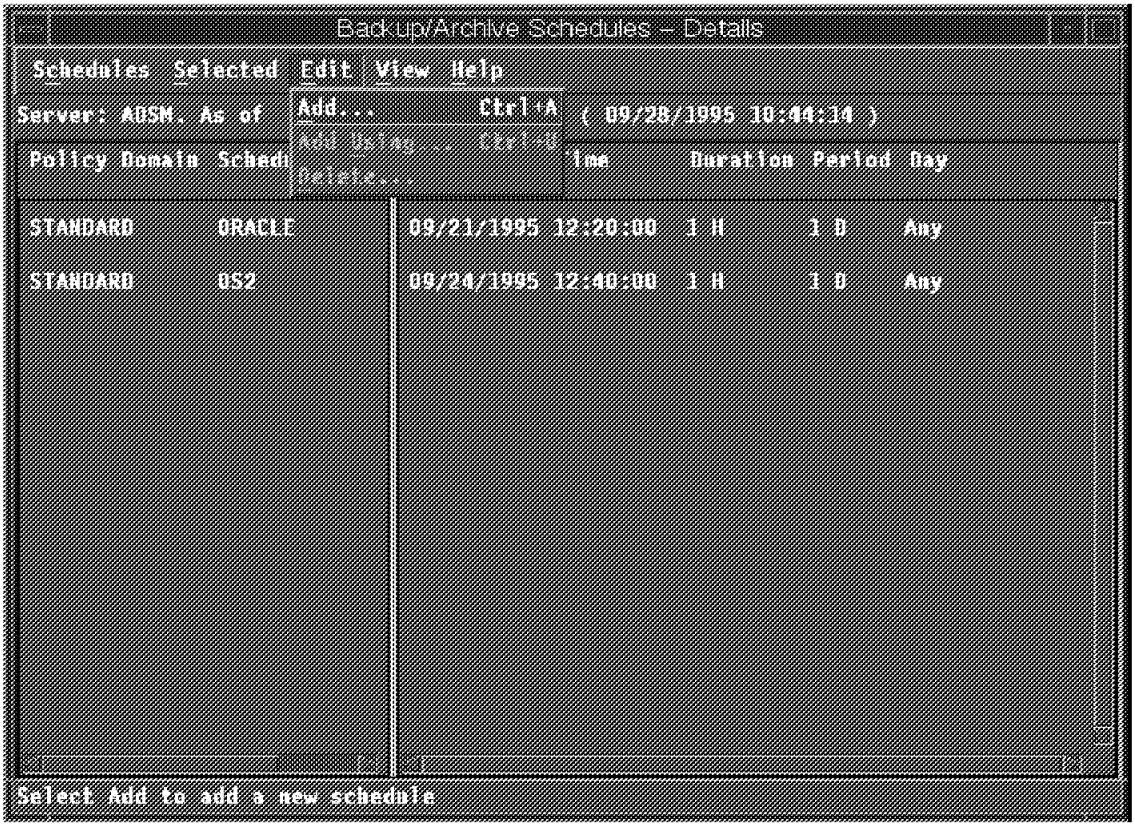


Figure 33. Add a Backup/Archive Scheduler Event

Select a policy domain, enter a name and description for the schedule, and add the nodes on which to execute this schedule by clicking on *Add nodes...* and including the list of node names (not shown). Then select the Operation notebook tab. The operation notebook page appears as shown in Figure 34 on page 75.

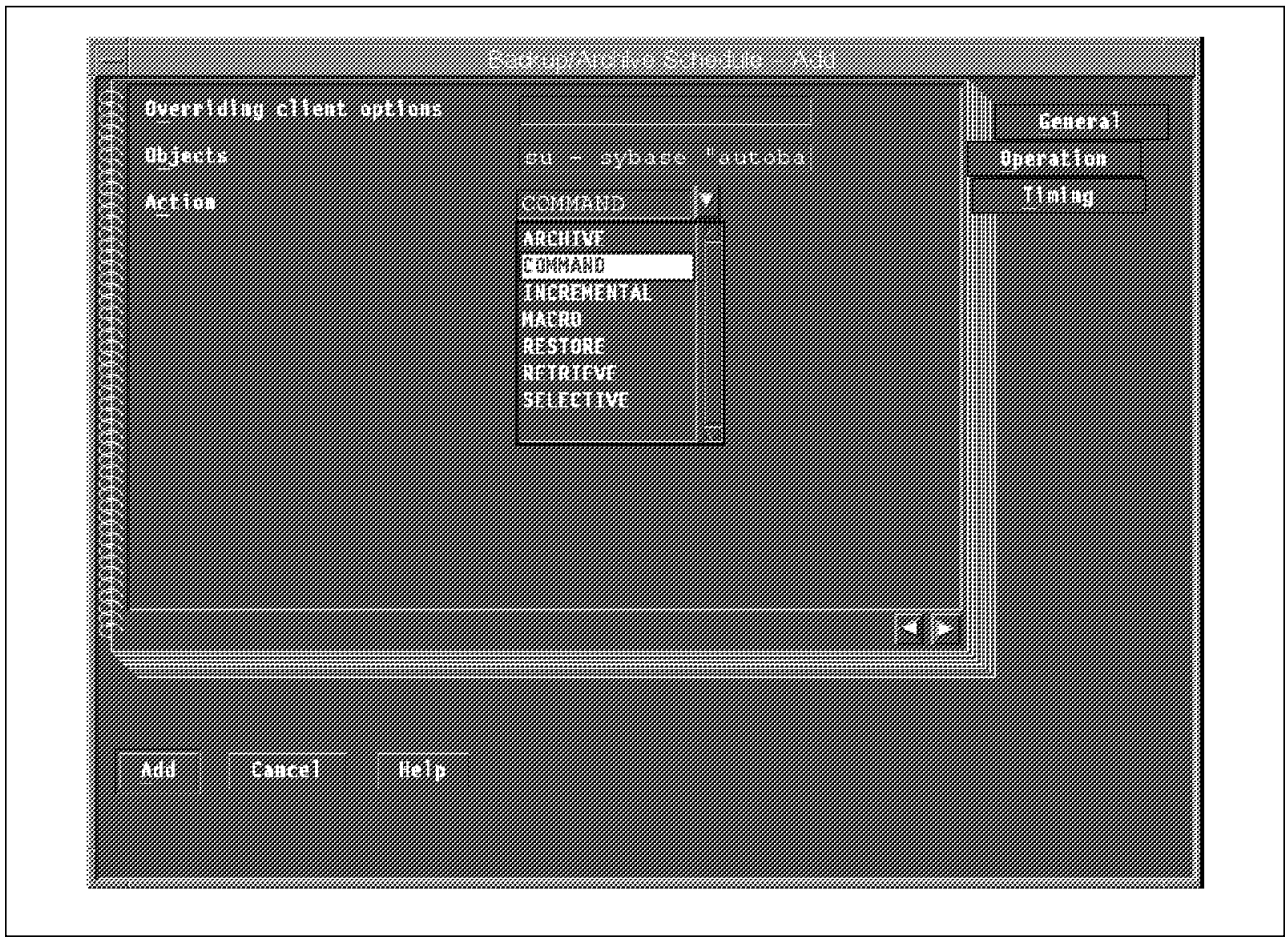


Figure 34. Backup/Archive Schedule Operation Notebook Page

Select action type COMMAND and enter, in the Objects entry field, the command you want the scheduler to execute on the client. In our example, we enter **su - sybase -c "autoback"**. Finally, select the Timing notebook tab to specify when you want the scheduler event to run.

The ADSM scheduler daemon must be running on your client for the scheduler event to be triggered correctly. To start the daemon use the **dsmc schedule** client command or click on the client scheduler icon. You can run the scheduler daemon in the background if necessary, but you might find it useful to see the output to ensure that your automation works.

Figure 35 on page 76 shows the output after you start the scheduler daemon on the client.

```
# dsmc schedule
ADSTAR Distributed Storage Manager
Command Line Backup Client Interface - Version 2, Release 1, Level 0.0
(C) Copyright IBM Corporation, 1990, 1995, All Rights Reserved.

Session established with server ADSM: AIX-RS/6000
  Server Version 2, Release 1, Level 0.1
  Data compression forced off by the server
  Server date/time: 09/11/1995 14:35:24   Last access: 09/11/1995 14:35:24

Querying server for next scheduled event.
Next operation scheduled:
------------------------------------------------------------
Schedule Name:         SYBASE
Action:                Command
Objects:               su - sybase -c 'autoback'
Options:
Server Window Start:   14:30:00  on 09/11/1995
------------------------------------------------------------
Command will be executed in 6 minutes.
```

Figure 35. Output of the Startup of the ADSM Scheduler Daemon on the Client

Figure 36 on page 77 shows the results of the execution of the scheduled event. Notice at the end of the process that the scheduler sends back details of the success or failure of the event to the central scheduler. An operator can therefore identify those events that failed.

```
when the Schedulre Event Executes
Executing scheduled command now.

Executing Operating System command or script:
   su - sybase -c 'autoback'

Backup Server session id is:  31.  Use this value when executing the
'sp_volchanged' system stored procedure after fulfilling any volume change
request from the Backup Server.
Backup Server: 4.41.1.1: Creating new disk file /u/sybase/pubs2.dump.
Backup Server: 6.28.1.1: Dumpfile name 'pubs2952540CE9B  ' section number 0001
mounted on disk file '/u/sybase/pubs2.dump'
Backup Server: 4.58.1.1: Database pubs2: 704 kilobytes DUMPed.
Backup Server: 3.43.1.1: Dump phase number 1 completed.
Backup Server: 3.43.1.1: Dump phase number 2 completed.
Backup Server: 3.43.1.1: Dump phase number 3 completed.
Backup Server: 4.58.1.1: Database pubs2: 712 kilobytes DUMPed.
Backup Server: 3.42.1.1: DUMP is complete (database pubs2).
Backup Server session id is:  34.  Use this value when executing the
'sp_volchanged' system stored procedure after fulfilling any volume change
request from the Backup Server.
Backup Server: 4.41.1.1: Creating new disk file /u/sybase/pubs2.tr01.
Backup Server: 6.28.1.1: Dumpfile name 'pubs2952540CEA6  ' section number 0001
mounted on disk file '/u/sybase/pubs2.tr01'
Backup Server: 4.58.1.1: Database pubs2: 286 kilobytes DUMPed.
Backup Server: 4.58.1.1: Database pubs2: 290 kilobytes DUMPed.
Backup Server: 3.43.1.1: Dump phase number 3 completed.
Backup Server: 4.58.1.1: Database pubs2: 294 kilobytes DUMPed.
Backup Server: 3.42.1.1: DUMP is complete (database pubs2).
ADSTAR Distributed Storage Manager
Command Line Backup Client Interface - Version 2, Release 1, Level 0.0
(C) Copyright IBM Corporation, 1990, 1995, All Rights Reserved.

Selective Backup function invoked.

Session established with server ADSM: AIX-RS/6000
  Server Version 2, Release 1, Level 0.1
  Data compression forced off by the server
  Server date/time: 09/11/1995 14:41:47   Last access: 09/11/1995 14:35:24

Normal File-->         739,328 /u/sybase/pubs2.dump ..................... Sent
Normal File-->         311,296 /u/sybase/pubs2.tr01 ..... Sent
Selective Backup processing of '/u/sybase/pubs2.*' finished with 0 failures.

Finished command.  Return code is:
   0
Scheduled event 'SYBASE' completed successfully.
Sending results for scheduled event 'SYBASE'.
```

Figure 36. Output from the Scheduler Daemon on the Client

5.3 Automating OS/2 Backups

There is no standard way of scheduling anything under OS/2 as it does not have any time-driven programs. However, it is possible to write a simple REXX program to check the *time* and *date* periodically and test whether an event should occur. Other ready-to-use programs are available either as products or shareware. If you have access to the Internet, look at the Hobbes ftp site, **ftp://hobbes.nmsu.edu**, which has some demonstrations of products such as **chron**.

You can also use the ADSM central scheduler to trigger a REXX program on your client. This REXX program could issue commands to create a backup of your database on disk and then use ADSM commands to send the files to the ADSM server. This is the same approach as for AIX and was tested successfully.

The SystemView for OS/2 event scheduler can also be used to schedule any OS/2 command.

Part 2. Oracle Backup and Recovery with ADSM

Chapter 6. Oracle Backup Basics

This chapter describes how to back up Oracle databases with the ADSM backup-archive client. First we look at Oracle's database management system (DBMS) structure so that you understand which files exist and which files you need to factor in to your backup strategy. Then we look at the Oracle backup utilities. Next we show how to integrate ADSM into your backup strategy, with specific examples of the full, partial, online, offline, incremental, and table level backups that we performed. Oracle Version 7.2.2.3 was used for the examples in this chapter.

6.1 Oracle DBMS Structure

Figure 37 illustrates the key components of an Oracle7 or Oracle8 database that you must consider for backup and recovery are:

- The *system* tablespace
- User tablespaces
- Online redo logs
- Archived redo logs
- Control files
- Initialization parameter files
- Configuration files.

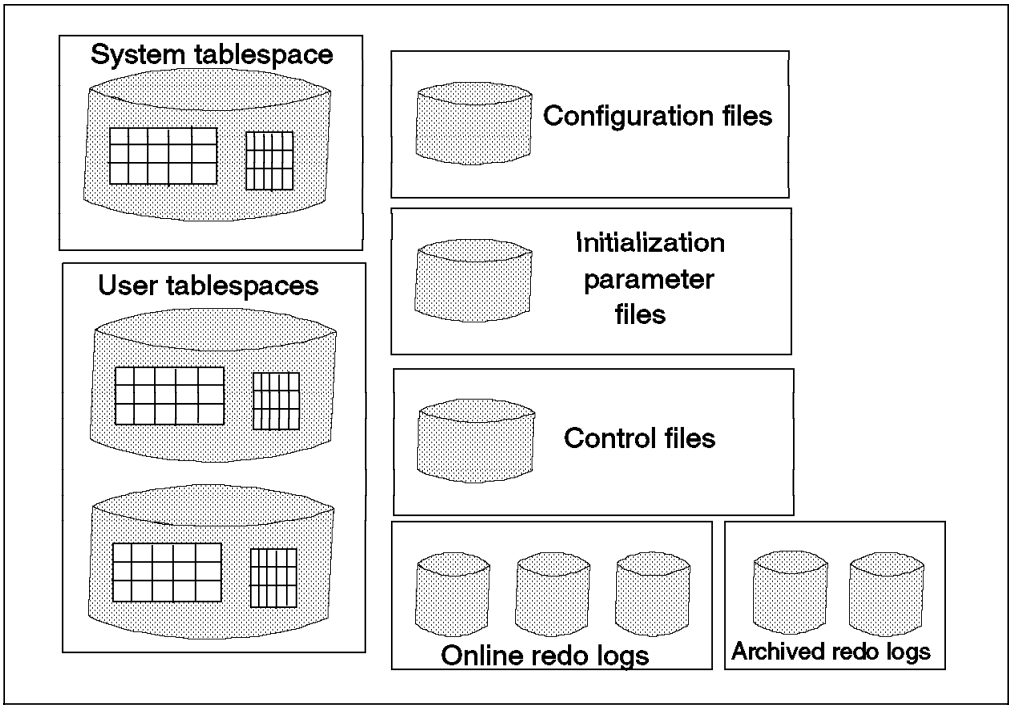


Figure 37. Oracle DBMS Structure

The **system tablespace** is automatically created when the database is created. It contains the data dictionary tables for the entire database. The data dictionary is a reference about the database and includes the names of the Oracle users,

users′ privileges, table names, space usage information, auditing information, and other general database information. The data dictionary is critical to the operation of the database because it records, verifies, and conducts ongoing work. The system tablespace is always online and cannot be taken offline because the data dictionary must always be available to Oracle. We recommend that you reserve the system tablespace for use by Oracle; do not create any user tables in this tablespace.

The **user tablespaces** contain all of the user data. This is where the bulk of the "real" database data is held.

The **redo log** is a set of files that record all changes made to the database so that in the event of a failure database updates are not lost. The redo log is comprised of two parts: the online redo log and the archived redo log. The **online redo log** consists of the current log files that are being used to record database changes. Optionally, filled online redo logs can be archived before being reused. If you have **archived redo logs** the database can be recovered from both instance and disk failures (instead of only instance failures), and the database can be backed up while it is open. To use archived redo logs, you run with the ARCHIVELOG mode set to on.

Control files, among other things, keep information about the physical structure of the database and log files. If you make a change to the database′s physical structure by adding, deleting, or renaming data or log files, you should back up the control files.

The **initialization parameter files** contain instance configuration parameters, such as how much memory to use and what to do with filled online redo logs.

Configuration files contain options that are common to multiple initialization parameter files.

Initialization parameter files and configuration files are always text files stored in the file system. They are not stored on raw devices.

For additional explanation of these various database structures, see 2.4, "Fundamentals of Relational Database Management Systems" on page 25. We recommend, however, that you:

- Use the LVM to mirror at least the disk files used for the system tablespace, control files, and online redo logs.
- Place at least one copy of the system tablespace, control files, and online redo logs on different physical volumes separate from the rest of the database.

6.2 Oracle Backup Utilities

Oracle does not provide its own backup utilities with the base product; instead it provides a command, **alter tablespace**, that aids in backups performed by operating system commands or other products (such as ADSM). Oracle also provides **export/import** commands that are designed to move Oracle data but can also be used to supplement backups. Oracle does provide a backup command to back up its control file: **alter database backup controlfile**.

In addition to these commands there are three additional products to back up and restore Oracle databases:

- Oracle7 Enterprise Backup Utility (EBU) provided by Oracle
- Oracle8 Recovery Manager (RMAN) provided by Oracle
- SQL-BackTrack provided by BMC Software, Inc.

All three products can back up directly to ADSM using the ADSM API; no intermediate file is created. Oracle can be installed on raw devices or file system files.

6.2.1 Alter Tablespace

There are two alternatives for using the alter tablespace command. Alter tablespace provides a way of taking a tablespace offline while the rest of the database remains in use and a way of backing up a tablespace while it is being updated (online). Forward recovery (using redo logs) is provided on restored tablespaces. You can think of alter tablespace as a way of “freezing” the tablespace such that the database has knowledge of where the backup begins. While a tablespace is in backup mode all changes to that tablespace are recorded in the online and archived redo logs. The changes during an online backup of a tablespace are not saved by backing up that tablespace. Furthermore, you must also back up the archived redo log files to back up the changes. Forward recovery provides consistency for any updates that occur after the tablespace is frozen.

To take backups for one or more tablespaces at a time, the database must be operating in ARCHIVELOG mode (see 6.1, “Oracle DBMS Structure” on page 81 for more information about this mode).

To take a backup of the tablespaces while they are offline, enter the **alter table space offline** command, followed by the backup commands from the backup utility (such as ADSM). Then enter the **alter tablespace online** command.

We show examples of both offline and online tablespace backups using the alter tablespace commands with ADSM in 6.3, “Using ADSM to Back Up Oracle” on page 86. This alter tablespace approach is supported with ADSM only when Oracle is installed on file system files. Using the alter tablespace approach for databases installed on raw devices requires the unsupported “as-is” ADSMPIPE utility (see Appendix C, “ADSMPIPE: Raw Logical Volume Backups with the API” on page 553 for details).

6.2.2 Export/Import

Oracle provides a pair of commands for exporting and importing data. These invoked from the AIX prompt using the **exp** and **imp** commands, respectively. You can use these to export and import:

- An entire database
- All of the database objects owned by a particular user
- Individual tables.

You also can use the **exp** and **imp** commands to export and import database objects incrementally. After entering the export command and directing the output to a file, you then use ADSM commands to back up that file to ADSM storage.

You can also use SQL-BackTrack to export directly to ADSM without creating an intermediate file (see Chapter 9, “SQL-BackTrack” on page 229).

Although you use export while the database is open and available for use, if you use export as a means of backup, all data must be exported in a logically consistent way so that the backup reflects a single point in time. No one should make changes to the database while the export takes place unless you use the *consistent=y* option. The *consistent=y* option of the Oracle **exp** command causes all changes made to the database tables during the export process to be tracked in rollback segments. Rollback segments are Oracle internal tables that hold information about uncommitted transactions. All changes are applied to the tables after the export process completes. Therefore, changes to database objects that occur during the export process are not reflected in the export output file. Ideally, you should run the database in exclusive mode during the export so that regular users cannot access the data. Use the Oracle *startup open exclusive* command to start up a database in exclusive mode. Although the mode of the database is online, it is not really online because connected users cannot use the database tables.

If an exclusive mode of the database is not possible, perform a consistent export with the *consistent=y* option.

We show examples of exporting the entire database and a single table, and then backing up the resultant export files with ADSM in 6.3, “Using ADSM to Back Up Oracle” on page 86.

6.2.3 Control File Backup

You need to back up, at a minimum, the control file of a database after making a structural modification (for example, adding a tablespace) to the database. You can use the **alter database backup controlfile** command to back up the control file to a file system file and ADSM commands to back up the file system file. The database must be operating in ARCHIVELOG mode.

Please note that in 6.3, “Using ADSM to Back Up Oracle” on page 86, we explain how to back up the control files, as well as the system tablespace, initialization parameter files, and configuration files, using another alternative: ADSM backup of the files directly without first using the **alter database backup controlfile** command.

Note: In an online backup, control files should be backed up last, because they contain header information for data files. Header information is required to recover the database to the most recent point in time.

6.2.4 EBU

EBU, previously known as the Oracle Parallel Backup/Restore utility or OBACKUP, is Oracle′s backup utility for Oracle7 databases. It provides functions to perform online and offline backups of the database, tablespaces, individual data files, control files, and redo logs. Restore and recovery can be performed for any component of the database. Roll forward recovery is performed by restoring redo logs and applying them to the restored database to the desired point in time.

EBU is a separate, installable product supplied with the Oracle7 Server product at no additional cost. EBU works in conjunction with ADSM. The interface to ADSM is provided by the ADSMConnect Agent for Oracle. Because the

ADSMConnect Agent for Oracle Backup application uses the ADSM API, an intermediate file is not created; the data is sent directly to ADSM.

Chapter 7, “Oracle7 Enterprise Backup Utility” on page 105 explains how to use EBU and ADSM to back up and restore an Oracle7 database.

6.2.5 RMAN

RMAN is Oracle′s backup utility for Oracle8 databases. It provides functions to perform online and offline backups of the database, tablespaces, individual data files, control files, and redo logs. In addition it provides the capability to perform incremental database backups. These are backups of only those data blocks within data files that have changed data since the previous incremental backup. Restore and recovery can be performed for any component of the database. Roll forward recovery is performed by restoring the appropriate combination of incremental backups and redo logs and applying them to the restored database to the desired point in time.

RMAN is part of the Oracle8 Server product and is automatically installed with it. RMAN works in conjunction with ADSM. The interface to ADSM is provided by the ADSMConnect Agent for Oracle. Because the ADSMConnect Agent for Oracle Backup application uses the ADSM API, an intermediate file is not created; the data is sent directly to ADSM.

Chapter 8, “Oracle8 Recovery Manager” on page 155 explains how to use RMAN and ADSM to back up and restore an Oracle8 database.

6.2.6 SQL-BackTrack

SQL-BackTrack is BMC Software, Inc.′s complete database backup solution for Oracle, Sybase, and Informix databases. For Oracle7 and Oracle8 databases, it provides functions to perform online and offline backups of the database, tablespaces, individual data files, control files, and redo logs. Online database backups can be either full or incremental for both Oracle7 and Oracle8. This ability to perform true incremental backups of Oracle7 databases is unique to SQL-BackTrack. SQL-BackTrack uses the ADSM API to store database objects directly within the ADSM server; an intermediate file is not created.

Chapter 9, “SQL-BackTrack” on page 229 explains how to use SQL-BackTrack and ADSM to back up and restore an Oracle database.

6.2.7 SAP/R3 Oracle

Oracle7 is also a popular choice of database for SAP/R3 environments. SAP/R3 provides backup and recovery commands for administrators. These commands can be used to perform online or offline backups of the database, tablespace, individual data files, control files, and redo logs. The backups are sent to a SAP backup interface, BACKINT. IBM provides a SAP backup interface module called BACKINT/ADSM. BACKINT/ADSM uses the ADSM API to store the database backup objects on one or more ADSM servers.

Chapter 10, “SAP R/3 Oracle and BACKINT/ADSM” on page 263 explains how to use BACKINT/ADSM to back up and restore an Oracle database in a SAP/R3 environment.

6.3 Using ADSM to Back Up Oracle

In the sections that follow we examine the ways in which you can use ADSM to back up Oracle databases. You can use several methods to back up and restore an Oracle database:

- Use ADSM to back up the Oracle files directly (if Oracle is on file system files).

  This type of backup is also called a *physical backup*, and it can save data files, tablespaces, redo log files, and control files.

- Use export to perform a logical backup of data.

In this section we assume that you use AIX file system files to store your database. The detailed examples we show are physical backups using ADSM commands directly:

- Partial offline backup and recovery
- Incremental offline backup and recovery
- Full offline backup and recovery
- Partial online backup and recovery
- Full online backup and recovery
- Full online (but not in use) export and import

The above list is not complete; there are other alternatives for backing up and restoring an Oracle database.

When the database is installed on raw devices, an alternative method must be used back up the Oracle files. In this situation, export, EBU with Oracle7, RMAN with Oracle8, or SQL-BackTrack for Oracle7 and Oracle8 must be used. Another alternative for raw devices is the ADSMPIPE as-is utility as described in Appendix C, “ADSMPIPE: Raw Logical Volume Backups with the API” on page 553. However, the best alternatives are EBU, RMAN, or SQL-BackTrack.

The examples to back up and restore an Oracle7 database with EBU are shown in Chapter 7, “Oracle7 Enterprise Backup Utility” on page 105.

The examples to back up and restore an Oracle8 database with RMAN are shown in Chapter 8, “Oracle8 Recovery Manager” on page 155.

The examples to back up and restore an Oracle7 or Oracle8 database with SQL-BackTrack are shown in Chapter 9, “SQL-BackTrack” on page 229.

6.3.1 Directory Structure of Our Sample Oracle Database

Figure 38 on page 87 shows the AIX directory structure of the Oracle database we use in our examples.

```
/oracle                          Oracle home directory

/oracle/dbs                      Oracle profiles

/oracle/dbs/systorfs.dbf         Oracle system tablespace
/oracle/dbs/rbsorfs.dbf          Oracle rollback segments tablespace
/oracle/dbs/temporfs.dbf         Oracle temporary segments tablespace
/oracle/dbs/toolorfs.dbf         Oracle tools tablespace
/oracle/dbs/usrorfs.dbf          Oracle user tablespace

/oracle/dbs/log1orfs.dbf         Oracle online redo log 1
/oracle/dbs/log2orfs.dbf         Oracle online redo log 2
/oracle/dbs/log3orfs.dbf         Oracle online redo log 3

/oracle/dbs/ctrl1orfs.ctl        Oracle first copy of control file
/oracle/dbs/ctrl2orfs.ctl        Oracle second copy of control file
/oracle/dbs/ctrl3orfs.ctl        Oracle third copy of control file

/oracle/orfs/archlogs            Oracle archived redo logs
```

Figure 38. AIX Directory Structure of Our Sample Oracle Database

For our examples and tests, we did not follow Oracle's recommended standard structure for a database, the *Oracle Flexible Architecture (OFA)*. We followed the Oracle menus of the Oracle *orainst* installation utility and accepted the prompted default file names and sizes when creating our sample Oracle database. We strongly recommend that you follow the Oracle database design considerations, as described in the *Oracle7 Server Administrator's Guide*, for performance and security reasons. For example, in a real customer environment you would not store all three copies of the control file in the same file system!

6.4 Partial Offline Backup and Recovery with ADSM

A partial offline backup backs up a subset of the database files, for example, the files that constitute a table space.

You may have a large database that consists of multiple table spaces, some of which frequently change and others that rarely change. At first sight, it may appear that the best approach would be to shut down the database and back up the files that make up the busiest table spaces. However, this technique can result in corruption of the database, and we advise against its use.

An alternative, safer technique involves keeping the database online but taking one table space offline. This is the technique we recommend and show here.

6.4.1 Preparatory Steps

Before you can use ADSM to perform a partial offline backup of an Oracle database, you must first compile a list of the names of the files that you want to back up. This list should consist of:

- Control files
- Data files
- Redo logs
- Parameter files

To compile the necessary information, perform the following steps:

1. Log in as the Oracle administrator and start SQL*DBA. Then connect to the database:

```
login oracle
sqldba lmode=y
  connect internal;
```

2. List the names of the files to be backed up:
   - Database control files

```
show parameter control_files;
```

   - Data files

```
select name from sys.v$datafile;
```

   - Redo logs

```
select member from sys.v$logfile;
```

   - Parameter files

     There is no simple way of listing all parameter files. You may find it useful to keep all of the parameter files for a database in a single directory. This will help you locate, list, and examine all of the parameter files that can be used with your database.

SQL*DBA

These examples were performed on an Oracle 7.2.2.3 database that supported SQL*DBA. Oracle 7.3 and above no longer support SQL*DBA. The Oracle server manager can be used to perform similar functions such as connecting to Oracle instances.

6.4.2 Backing Up a Tablespace Using SQL*DBA Line Mode

If you want to perform a backup of a tablespace at the SQL*DBA command line, execute the following Oracle and ADSM commands:

1. Log in to AIX and start SQL*DBA in line mode:

```
login oracle
sqldba lmode=y
```

2. Connect to the database, take the USERS tablespace offline, and exit from SQL*DBA:

```
  connect internal;
  alter tablespace users offline normal;
exit;
```

3. Use the ADSM dsmc archive command from the AIX command line to back up the tablespace file, */oracle/dbs/usrorfs.dbf*:

```
dsmc archive -de="ORFS USERS 1995-09-18" /oracle/dbs/usrorfs.dbf
```

4. Start SQL*DBA, connect to the database, and set the USERS tablespace online, after ADSM successfully completes the dsmc archive command:

```
sqldba lmode=y
  connect internal;
  alter tablespace users online;
  exit;
```

6.4.3 Restoring a Tablespace

If a media failure damages one or more of the files that constitute the tablespace, you can recover the tablespace without having to recover the entire database. Follow these steps:

1. Log in to AIX as the Oracle administrator, start the SQL*DBA command line session, and connect to the database:

```
login oracle
sqldba lmode=y
  connect internal;
```

2. Take the tablespace USERS offline and exit SQL*DBA:

```
  alter tablespace users offline;
  exit;
```

3. Issue the ADSM dsmc retrieve command from the AIX command line session to retrieve the /oracle/dbs/usrorfs.dbf data file of tablespace USERS:

```
  dsmc retrieve -desc="ORFS USERS 1995-09-18" /oracle/dbs/usrorfs.dbf
```

4. After ADSM successfully completes the retrieve operation, return to the SQL*DBA session, connect to the databases, and perform media recovery on the tablespace:

```
sqldba lmode=y
  connect internal;
  recover tablespace users;
```

5. Bring the tablespace back online:

```
  alter tablespace users online;
```

6. Verify that the tablespace is online:

```
  select * from sys.v$datafile;
```

When you use ADSM to directly back up file system files, you must use the MAKESParsefile=no option on the restore or retrieve command. See 4.3.1, “Handling Sparse Files” on page 56 for more information about sparse files.

6.5 Incremental Offline Backup with ADSM

You may have a large database that contains data that is frequently updated. You may want to back up the database frequently, perhaps several times per day. However, the period of time during which the database can be offline may be insufficient for you to perform a full backup. In such circumstances you can perform an incremental offline backup.

The technique we recommend involves using ADSM to back up Oracle's *archived redo logs*.

Note: Ensure that you have a full offline backup of the entire database.

6.5.1 Preparatory Steps

To perform incremental offline backups of an Oracle database, you must ensure that Oracle will archive its redo logs to AIX file system files, by starting the automatic archive process.

Start SQL*DBA in line mode, connect to the database, and start the automatic archiving process:

```
sqldba lmode=y
connect internal;
archive log start;
exit;
```

You can check to see whether Oracle is archiving its logs to file system files by connecting to the database with DBA authority. Then select the *Log* item from the action bar, followed by the *List Archive Status* option. The database log mode will show as **ARCHIVELOG**, and the automatic archival option will show as **ENABLED**.

The destination to which archived logs are written is specified by the *log_archive_dest* keyword in the initialization parameter files.

The following AIX cat command shows you the destination of the archived redo logs by returning a line of the Oracle *configorfs.ora* file:

```
$ cat /oracle/dbs/configorfs.ora | grep log_archive_dest
log_archive_dest                  = /oracle/orfs/archlogs/arch
```

In our example, the redo logs are archived to a file system with the following file names:

```
/oracle/orfs/archlogs/arch1_51.dbf
/oracle/orfs/archlogs/arch1_52.dbf
/oracle/orfs/archlogs/arch1_53.dbf
```

6.5.2 Backing Up the Archived Redo Logs

We recommend that you use the ADSM dsmc incremental command to back up the archived redo logs. Because every redo log has a unique file name, using an ADSM incremental backup will ensure that you back up every redo log:

```
dsmc incremental /oracle/orfs/archlogs
```

The advantage of using the ADSM dsmc incremental command is that you back up only those files that were created by Oracle since the last ADSM dsmc incremental process.

To check that the backup has been successful for all archived redo logs use the following command:

```
dsmc query backup "/oracle/orfs/archlogs/arch*"
```

The ADSM dsmc query backup command returns the files shown in Figure 39.

```
  Size      Backup Date       Mgmt Class A/I File
  ----      -----------       ---------- --- ----
20,480  09/28/1995 10:31:46   DEFAULT     A  /oracle/orfs/archlogs/arch1_14.dbf

28,160  09/28/1995 10:31:46   DEFAULT     A  /oracle/orfs/archlogs/arch1_15.dbf
```

Figure 39. Output of the ADSM dsmc query backup Command for Oracle

Generally, we recommend leaving as many archived redo logs as possible on disk so that they are available for recovery purposes. You can (optionally) delete the archived redo logs and release the disk space they occupy if you run out of disk space as shown below with the remove (rm) command. The only situation where you might want to delete the old archived redo logs from disk is after a full offline backup because it is unlikely that you will need them on disk for recovery.

```
rm -i /oracle/orfs/archlogs/arch*
```

You can also use ADSM's client HSM function to automatically and transparently migrate and recall the redo logs. HSM automatically clears up disk space by transparently storing the files on the ADSM server. The files are automatically recalled when accessed. You can use client HSM for any redo logs, not just the old archived redo logs after a full offline backup. See 4.3.6, "ADSM Hierarchical Storage Manager" on page 60 for a discussion on using client HSM for database log files.

6.5.3 Recovering the Database

If, as a consequence of a media failure, you have lost some or all of your database's tablespaces, you may decide to restore the tablespaces from a full backup that you took some time ago. Having done this, you may want to bring the database up to date by applying the changes that Oracle has recorded in the archived redo logs. The steps are:

1. Start SQL*DBA, connect to the database, and perform a shutdown of the database:

```
sqldba lmode=y
connect internal;
shutdown normal;
exit;
```

2. Use ADSM to retrieve the failed database object, for example, the tablespace USERS from the files that you archived earlier (as part of a full offline backup, for example).

   From the AIX command line issue:

```
dsmc retrieve -desc="ORFS USERS 1995-09-27" /oracle/dbs/usrorfs.dbf
```

3. If you deleted the archived redo logs from disk, use ADSM to restore them before starting the database with the recovery option.

   Note: If you have to restore and recover just a single tablespace, do not restore the control files or the online redo logs. From the AIX command line issue:

```
dsmc restore "/oracle/orfs/archlogs/arch*"
```

4. Start SQL*DBA and connect to the database:

```
sqldba lmode=y
connect internal;
```

5. Start up the database in mount mode and enable automatic recovery:

```
startup mount;
set autorecovery on;
```

6. Shut down the database and start up the database in recovery mode:

```
shutdown normal;
startup open recover;
exit;
```

When you use ADSM to directly back up file system files, you must use the MAKESParsefile=no option on the restore or retrieve command. See 4.3.1, "Handling Sparse Files" on page 56 for more information about sparse files.

6.6 Full Offline Backup and Recovery with ADSM

It is important to recognize that full offline backup and recovery enables you to restore a database to the state it was in when the backup was taken. You cannot update the database with changes that have been made since the backup was taken. However, you can apply archived redo logs to roll forward the database to a more recent point in time.

Here we show how to set up and perform a physical offline backup and restore.

6.6.1 Preparatory Steps

Before you perform a full, offline backup of an Oracle database, you must compile a list of the data files that each tablespace uses (see 6.4.1, “Preparatory Steps” on page 87).

You can perform all Oracle backup and restore operations from either the SQL*DBA GUI or the SQL*DBA command line. In our examples we use the SQL*DBA command line and Oracle commands. Use the SQL*DBA GUI if you are not familiar with Oracle commands. Use the SQL*DBA command line if you are familiar with Oracle commands or want to create backup scripts.

6.6.2 Backing Up the Database

The steps to back up the database are:

1. Log in to AIX as the Oracle user, start SQL*DBA in line mode, and connect to the database:

```
login oracle
sqldba lmode=y
  connect internal;
```

2. Shut down the database and exit SQL*DBA:

```
  shutdown normal;
  exit;
```

3. Prepare a list of Oracle data files, control files, and redo log files from the AIX command line. Change your AIX directory to the path where the Oracle data files are located. Export the list of data file names to a temporary AIX environment variable:

```
  cd /oracle/dbs
  export ORFS_FILES=ls *.dbf
```

4. Use ADSM to archive the list of files with a unique description:

```
  dsmc archive -de="ORFS 1995-09-28" $ORFS_FILES
```

   Note: We recommend using the ADSM *archive* command for full, offline backup because it eases the task of identifying the full set of files that have to be restored. See 4.3.3, “Grouping Related Files” on page 59 for a more detailed explanation.

5. Use ADSM to archive the parameter files and control files:

```
  dsmc archive -de="ORFS 1995-09-28" /oracle/dbs/initorfs.ora
  dsmc archive -de="ORFS 1995-09-28" /oracle/dbs/configorfs.ora
  dsmc archive -de="ORFS 1995-09-28" "/oracle/dbs/ctrl*orfs.ctl"
```

6. Optionally, back up archived redo logs files, but understand that all of the information in the log files is contained in the data files for a full, offline backup. Use the ADSM dsmc incremental command to back up the archived redo logs of the AIX */oracle/orfs/archlogs* file system:

```
dsmc incr /oracle/orfs/archlogs
```

7. Check that the archives were successful:

```
dsmc query archive "/oracle/dbs/*" | grep ORFS
```

Figure 40 shows the results of the ADSM dsmc query archive command.

```
       707  09/28/1995 22:36:35   /oracle/dbs/configorfs.ora 09/27/1996 ORFS
   111,104  09/28/1995 22:36:43   /oracle/dbs/ctrl1orfs.ctl 09/27/1996 ORFS
   111,104  09/28/1995 22:36:48   /oracle/dbs/ctrl2orfs.ctl 09/27/1996 ORFS
   111,104  09/28/1995 22:36:52   /oracle/dbs/ctrl3orfs.ctl 09/27/1996 ORFS
     4,617  09/28/1995 22:36:28   /oracle/dbs/initorfs.ora 09/27/1996 ORFS
 1,049,088  09/28/1995 22:34:11   /oracle/dbs/log1orfs.dbf 09/27/1996 ORFS
 1,049,088  09/28/1995 22:34:24   /oracle/dbs/log2orfs.dbf 09/27/1996 ORFS
 1,049,088  09/28/1995 22:34:28   /oracle/dbs/log3orfs.dbf 09/27/1996 ORFS
 8,392,704  09/28/1995 22:34:41   /oracle/dbs/rbsorfs.dbf 09/27/1996 ORFS
41,947,136  09/28/1995 22:35:35   /oracle/dbs/systorfs.dbf 09/27/1996 ORFS
 1,052,672  09/28/1995 22:35:38   /oracle/dbs/temporfs.dbf 09/27/1996 ORFS
26,218,496  09/28/1995 22:36:09   /oracle/dbs/toolorfs.dbf 09/27/1996 ORFS
 1,052,672  09/28/1995 22:36:13   /oracle/dbs/usrorfs.dbf 09/27/1996 ORFS
```

Figure 40. Output of the ADSM dsmc query archive Command for Oracle

8. Start SQL*DBA in line mode and connect to the database:

```
sqldba lmode=y
  connect internal;
```

9. Start up the database:

```
  startup open;
```

6.6.3 Restoring the Database

The steps to restore the database are:

1. Log in to AIX as the Oracle user, start SQL*DBA, connect, and shut down the database:

```
login oracle
sqldba lmode=y
  connect internal;
  shutdown normal;
  exit;
```

2. Go to the AIX command line to retrieve the files from ADSM; use either the ADSM GUI or commands to identify and retrieve the archived files:

```
  dsmc retrieve -de="ORFS 1995-09-28" "*.dbf"
```

3. Use ADSM to retrieve the parameter and control files:

```
dsmc retrieve -de="ORFS 1995-09-28" /oracle/dbs/initorfs.ora
dsmc retrieve -de="ORFS 1995-09-28" /oracle/dbs/configorfs.ora
dsmc retrieve -de="ORFS 1995-09-28" "/oracle/dbs/ctrl*orfs.ctl"
```

4. Restore the archived redo logs files, if needed. Use the ADSM dsmc restore command to restore the archived redo logs of the AIX /oracle/orfs/archlogs file system:

```
dsmc restore /oracle/orfs/archlogs
```

5. Return to the SQL*DBA session, connect, and start up the database:

```
sqldba lmode=y
  connect internal;
  startup open;
  exit;
```

When you use ADSM to directly back up file system files, you must use the MAKESParsefile=no option on the restore or retrieve command. See 4.3.1, “Handling Sparse Files” on page 56 for more information about sparse files.

6.7 Partial Online Backup and Recovery with ADSM

Your database may contain data that changes frequently and needs regular backup. However, the occasions on which you can take the database offline may be so rare or of such short duration that offline backup is not practical. In these circumstances partial online backup is appropriate. This process is similar to that discussed in 6.6, “Full Offline Backup and Recovery with ADSM” on page 92.

Note

The ADSM copy group serialization mode must be set to either dynamic or shared dynamic when you use the alter tablespace begin backup command. ADSM backs up a tablespace that is online and therefore might be in use.

6.7.1 Preparatory Steps

Before you can perform partial online backup of an Oracle database, you must compile a list of the data files that each tablespace uses.

6.7.2 Backing Up a Tablespace

The steps to back up a tablespace are:

1. Log in to AIX as the Oracle administrator, start an SQL*DBA command line session, and connect to the database:

```
login oracle
sqldba lmode=y
  connect internal;
```

2. You must indicate to Oracle that you are about to start online backup of the tablespace and exit SQL*DBA:

```
  alter tablespace users begin backup;
  exit;
```

3. From the AIX command line issue the ADSM command to archive the USERS tablespace data files (we use the *-de* option to identify the archive):

```
  dsmc archive -de="ORFS USERS 1995-09-18" /oracle/dbs/usrorfs.dbf
```

4. When ADSM has successfully completed the backup, return to the SQL*DBA session and indicate to Oracle that you have completed the backup:

```
sqldba lmode=y
  connect internal;
  alter tablespace users end backup;
  exit;
```

6.7.3 Restoring a Tablespace

If a media failure damages one or more of the files that constitute the tablespace, you can recover the tablespace without having to recover the entire database. Follow these steps:

1. Log in to AIX as the Oracle administrator, start an SQL*DBA command line session, and connect to the database:

```
login oracle
sqldba lmode=y
  connect internal;
```

2. Take the tablespace offline and exit SQL*DBA:

```
  alter tablespace users offline;
  exit;
```

3. From the AIX command line invoke the ADSM dsmc retrieve command to retrieve the USERS tablespace, /oracle/dbs/usrorfs.dbf, which is in the data file:

```
  dsmc retrieve  -de="ORFS USERS 1995-09-18" /oracle/dbs/usrorfs.dbf
```

4. When ADSM has successfully completed the restore, return to the SQL*DBA session and perform media recovery on the tablespace:

```
sqldba lmode=y
  connect internal;
  recover tablespace users;
```

5. Bring the tablespace back online:

```
  alter tablespace users online;
```

6. Verify the status of all tablespaces:

```
  select * from sys.v$datafile;
```

When you use ADSM to directly back up file system files, you must use the MAKESParsefile=no option on the restore or retrieve command. See 4.3.1, “Handling Sparse Files” on page 56 for more information about sparse files.

6.8 Full Online Backup and Recovery with ADSM

This example shows you how you can take a full online backup while the database is in an open mode and in use.

6.8.1 Backing Up All of the Tablespaces and Data Files

The approach with this example is to use a shell script to perform the backup. To start the script use one of the scheduling methods shown in 5.2.4, “Using the ADSM Central Scheduler” on page 73, because it is likely that you would want to perform this backup unattended.

Note: Start the backup shell script as the AIX root user and switch to Oracle user during execution. Starting as the AIX root user sets your environment correctly, and you are not prompted for a password when you become the Oracle user.

You must compile a list of file names you want to back up when you perform a full backup of the database. If your database structure changes very frequently such that you have different data file and path names, we recommend that you compile the list of files dynamically and export it temporarily through an AIX environment variable, as shown in 6.6.2, “Backing Up the Database” on page 93. If your database structure does not change very often such that you always have the same data file and path names, we recommend that you use predefined files with absolute path names in the shell script.

Note

The ADSM copy group serialization mode must be set to either dynamic or shared dynamic when you use the alter tablespace begin backup command. ADSM backs up a tablespace that is online and therefore might be in use.

The steps you perform are the same as those for a partial online backup operation:

1. Indicate to Oracle that you are going to start an online backup of a tablespace. Use the Oracle *ALTER TABLESPACE BEGIN BACKUP* command.
2. Use ADSM to archive all of the data files of all of the tablespaces with a unique description.
3. Indicate to Oracle that you have completed the online backup of the tablespace. Use the Oracle *ALTER TABLESPACE END BACKUP* command.

   Repeat steps 1 through 3 for each tablespace. Database users might experience better performance if you repeat steps 1 through 3 in sequence for each tablespace instead of issuing step 1 for all tablespaces, step 2 for all tablespaces, and step 3 for all tablespaces.
4. Switch the current online redo logs to convert them to archived redo logs by using the Oracle *ALTER SYSTEM SWITCH LOGFILE* command and determine which archived redo logs you have to back up. You will want to repeat this step more than once, according to the amount of online redo log files that are currently in use. The best way to perform this step unattended in a shell script is to issue the Oracle *ALTER SYSTEM SWITCH LOGFILE* command as many times as the number of online redo log files you have defined. Subsequently, all online redo log files will be switched, and you ensure that you back up all online redo logs currently in use.

5. Use ADSM to back up the archived redo logs. Use the ADSM dsmc incremental command.
6. Back up the control file to a file system file and use ADSM to archive the control file with the same description as the tablespace data files. Use the Oracle *ALTER DATABASE BACKUP CONTROLFILE* command. We recommend backing up the control files last because the control files store header information for data files. The database needs this information to recover the data files.
7. Use ADSM to archive the Oracle parameter files and, optionally, other Oracle-specific files you want to save (such as profiles, export files, and SQL scripts).
8. To verify that your shell script works without any ADSM failures, check your backup log file, *backup.log*, which stores the output of the shell script. On the AIX command line issue:

```
cat backup.log | grep failure
```

Figure 41 shows the output of the AIX cat command.

```
Archive processing of '/oracle/dbs/systorfs.dbf' finished with 0 failures.
Archive processing of '/oracle/dbs/rbsorfs.dbf' finished with 0 failures.
Archive processing of '/oracle/dbs/usrorfs.dbf' finished with 0 failures.
Archive processing of '/oracle/dbs/temporfs.dbf' finished with 0 failures.
Archive processing of '/oracle/dbs/toolorfs.dbf' finished with 0 failures.
Archive processing of '/oracle/dbs/log1orfs.dbf' finished with 0 failures.
Archive processing of '/oracle/dbs/log2orfs.dbf' finished with 0 failures.
Archive processing of '/oracle/dbs/log3orfs.dbf' finished with 0 failures.
Archive processing of '/oracle/control.back' finished with 0 failures.
Archive processing of '/oracle/dbs/ctrl1orfs.ctl' finished with 0 failures.
Archive processing of '/oracle/dbs/ctrl2orfs.ctl' finished with 0 failures.
Archive processing of '/oracle/dbs/ctrl3orfs.ctl' finished with 0 failures.
Archive processing of '/oracle/dbs/init*' finished with 0 failures.
Archive processing of '/oracle/dbs/config*' finished with 0 failures.
```

Figure 41. Output of the AIX cat Command for Oracle

Your ADSM commands are successful if every ADSM command returns with a “finished with 0 failures” message.

Figure 42 shows our sample shell script for full online backup.

```
#!/bin/ksh
#--------------------------------------------------------------------
#  Script:  full_oline_backup.ksh
#  Performs a full, online backup of an entire Oracle database
#--------------------------------------------------------------------
export DATE=date
export DSMI_CONFIG=/oracle/dsm.opt
export DSMI_DIR=/usr/lpp/adsm/bin
export DSMI_LOG=/oracle
#--------------------------------------------------------------------
```

Figure 42 (Part 1 of 4). AIX Shell Script for Full Oracle Online Backup Using ADSM

```
#  Display status information at the beginning of the backup
#--------------------------------------------------------------------
su - oracle -c "sqldba lmode=y <<TAG
                connect internal;
                show parameters control_file;
                archive log list;
                select * from dba_data_files;
                e;
                TAG"
#--------------------------------------------------------------------
#  Start the backup of the tablespaces and use ADSM to archive the files
#--------------------------------------------------------------------
su - oracle -c "sqldba lmode=y <<TAG
                connect internal;
                alter tablespace system begin backup;
                e;
                TAG"

dsmc archive -de="FULL ONLINE $DATE" /oracle/dbs/systorfs.dbf

su - oracle -c "sqldba lmode=y <<TAG
                connect internal;
                alter tablespace system end backup;
                e;
                TAG"

su - oracle -c "sqldba lmode=y <<TAG
                connect internal;
                alter tablespace rbs begin backup;
                e;
                TAG"

dsmc archive -de="FULL ONLINE $DATE" /oracle/dbs/rbsorfs.dbf

su - oracle -c "sqldba lmode=y <<TAG
                connect internal;
                alter tablespace rbs end backup;
                e;
                TAG"

su - oracle -c "sqldba lmode=y <<TAG
                connect internal;
                alter tablespace users begin backup;
                e;
                TAG"

dsmc archive -de="FULL ONLINE $DATE" /oracle/dbs/usrorfs.dbf

su - oracle -c "sqldba lmode=y <<TAG
                connect internal;
```

Figure 42 (Part 2 of 4). AIX Shell Script for Full Oracle Online Backup Using ADSM

```
                alter tablespace users end backup;
                e;
                TAG"

su - oracle -c "sqldba lmode=y <<TAG
                connect internal;
                alter tablespace temp begin backup;
                e;
                TAG"

dsmc archive -de="FULL ONLINE $DATE" /oracle/dbs/temporfs.dbf

su - oracle -c "sqldba lmode=y <<TAG
                connect internal;
                alter tablespace temp end backup;
                e;
                TAG"

su - oracle -c "sqldba lmode=y <<TAG
                connect internal;
                alter tablespace tools begin backup;
                e;
                TAG"

dsmc archive -de="FULL ONLINE $DATE" /oracle/dbs/toolorfs.dbf

su - oracle -c "sqldba lmode=y <<TAG
                connect internal;
                alter tablespace tools end backup;
                e;
                TAG"
#--------------------------------------------------------------------
#  Switch the online redo log files
#--------------------------------------------------------------------
su - oracle -c "sqldba lmode=y <<TAG
                connect internal;
                alter system switch logfile;
                alter system switch logfile;
                alter system switch logfile;
                e;
                TAG"
#--------------------------------------------------------------------
#  Use ADSM to archive the online redo logs
#--------------------------------------------------------------------
dsmc archive -de="FULL ONLINE $DATE" /oracle/dbs/log1orfs.dbf
dsmc archive -de="FULL ONLINE $DATE" /oracle/dbs/log2orfs.dbf
dsmc archive -de="FULL ONLINE $DATE" /oracle/dbs/log3orfs.dbf
#--------------------------------------------------------------------
#  Use ADSM to backup the archived redo logs incrementally, which are
#  located in the file system /oracle/orfs/archlogs
```

Figure 42 (Part 3 of 4). AIX Shell Script for Full Oracle Online Backup Using ADSM

```
#--------------------------------------------------------------------
dsmc incr /oracle/orfs/archlogs
#--------------------------------------------------------------------
#  Save the controlfile to a directory /oracle/orfs
#--------------------------------------------------------------------
su - oracle -c "sqldba lmode=y <<TAG
                connect internal;
                alter database backup controlfile
                      to '/oracle/control.back' reuse;
                e;
                TAG"
#--------------------------------------------------------------------
#  Use ADSM to archive all control files
#--------------------------------------------------------------------
dsmc archive -de="FULL ONLINE $DATE"  /oracle/control.back
dsmc archive -de="FULL ONLINE $DATE" "/oracle/dbs/ctrl1orfs.ctl"
dsmc archive -de="FULL ONLINE $DATE" "/oracle/dbs/ctrl2orfs.ctl"
dsmc archive -de="FULL ONLINE $DATE" "/oracle/dbs/ctrl3orfs.ctl"
#--------------------------------------------------------------------
#  Use ADSM to archive other Oracle specific parameter files
#--------------------------------------------------------------------
dsmc archive -de="FULL ONLINE $DATE" "/oracle/dbs/initorfs*"
dsmc archive -de="FULL ONLINE $DATE" "/oracle/dbs/configorfs*"
#--------------------------------------------------------------------
#  Display status information at the end of the backup
#--------------------------------------------------------------------
su - oracle -c "sqldba lmode=y <<TAG
                connect internal;
                archive log list;
                e;
                TAG"
```

Figure 42 (Part 4 of 4). AIX Shell Script for Full Oracle Online Backup Using ADSM

You could recode the shell script to explicitly code a loop for backups of the tablespaces. The recoded shell script would be less error prone than the original shell script (Figure 42 on page 98) and could more easily be extended to include additional tablespaces. Here is an example of how to recode an explicit loop:

```
|foreach ts (system rbs users temp tools)
||su - oracle -c "sqldba lmode=y << TAG
|||connect internal;
|||alter tablespace $ts begin backup;
|||exit;
|||TAG"

||dsmc archive -de="FULL ONLINE $DATE" /oracle/dbs/${ts}orfs.dbf

||su - oracle -c "sqldba lmode=y << TAG
|||connect internal;
|||alter tablespace $ts end backup;
|||exit;
|||TAG"
|end
```

6.8.2 Restoring All Tablespaces and Data Files

Perform the following steps when you must restore the entire database because of loss of all tablespaces or inconsistencies in the database data:

1. Ensure that the database is shut down. If the database is not shut down, you must shut it down in abort mode.
2. Connect to SQL*DBA and shut down the database in abort mode:

```
sqldba lmode=y
connect internal;
shutdown abort;
exit;
```

3. Retrieve lost parameter files from ADSM.

   Retrieving the archived files from an ADSM server through the ADSM backup/archive client GUI is easier than retrieving by ADSM command line commands. At the ADSM GUI you can more easily search for archived files by entering the file description. You can easily select the files you want to restore. However, we want to show you the ADSM commands at the AIX command line.

   To retrieve the archived files by using ADSM commands, set the DATE environment variable of your AIX session to the date when the backup was taken. You can then use the environment variable in the description field of the ADSM dsmc retrieve command. From the AIX command line enter:

```
export DATE="Tue Sep 19 12:06:59 PDT 1995"
dsmc retrieve -de="FULL ONLINE $DATE" "/oracle/dbs/initorfs*"
dsmc retrieve -de="FULL ONLINE $DATE" "/oracle/dbs/configorfs*"
```

4. Retrieve all control files:

```
dsmc retrieve -de="FULL ONLINE $DATE" /oracle/control.back
dsmc retrieve -de="FULL ONLINE $DATE" "/oracle/dbs/ctrl1orfs.ctl"
dsmc retrieve -de="FULL ONLINE $DATE" "/oracle/dbs/ctrl2orfs.ctl"
dsmc retrieve -de="FULL ONLINE $DATE" "/oracle/dbs/ctrl3orfs.ctl"
```

   We recommend retrieving all control files to reflect the database structure as it was before the failure, although Oracle uses the backup copy of the control file to re-create the database structure and recovers using the header information stored in the control file.

5. Restore the archived redo log files:

```
dsmc restore "/oracle/orfs/archlogs/*"
```

6. Retrieve the data files:

```
dsmc retrieve -de="FULL ONLINE $DATE" /oracle/dbs/systorfs.dbf
dsmc retrieve -de="FULL ONLINE $DATE" /oracle/dbs/rbsorfs.dbf
dsmc retrieve -de="FULL ONLINE $DATE" /oracle/dbs/usrorfs.dbf
dsmc retrieve -de="FULL ONLINE $DATE" /oracle/dbs/toolorfs.dbf
dsmc retrieve -de="FULL ONLINE $DATE" /oracle/dbs/temporfs.dbf
```

7. Retrieve the online redo log files:

```
dsmc retrieve -de="FULL ONLINE $DATE" /oracle/dbs/log1orfs.dbf
dsmc retrieve -de="FULL ONLINE $DATE" /oracle/dbs/log2orfs.dbf
dsmc retrieve -de="FULL ONLINE $DATE" /oracle/dbs/log3orfs.dbf
```

Restoring online redo log files adds no value to the recovery process. Oracle overwrites the contents of the online redo log files with the contents of the archived redo log files during the recovery process. Nevertheless, we recommend that you restore the online redo log files to reflect the database structure as it was before the failure.

8. Connect to SQL*DBA and start up the database in mount mode:

```
sqldba lmode=y
connect internal;
startup mount;
```

9. Recover the database using the backup copy of the control file:

```
recover database using backup controlfile;
```

10. Accept the redo log file numbers from Oracle and press <RETURN> to continue. Oracle now recovers the database to the last recent point in time.
11. Shut down the database:

```
shutdown normal;
```

12. Start up the database in open mode:

```
startup open.
```

We recommend that you perform a full backup of the entire database after successfully recovering the database.

6.9 Full Online Export and Import

Exporting the contents of a database is not the recommended method of backing up a database, but it is useful for saving the specific contents of database objects, such as table definitions, table data, grants, triggers, views, and indexes. When exporting database objects, the database should ideally be in open exclusive mode. Therefore, an export in exclusive mode is an “almost offline” operation.

6.9.1 Exporting the Database

A full export of the entire database exports all database contents to file system files. The steps for a full export while the database is in exclusive online mode are:

1. Connect to the database. Shut down and start up the database in exclusive mode:

```
sqldba lmode=y
connect internal;
shutdown normal;
startup open exclusive;
exit;
```

2. Export the full database to a disk file.

   From the AIX command line issue this exp command:

```
exp userid=system/manager full=y file=/oracle/orfs.dmp
```

   Note: If you cannot shut down and restart the database in exclusive mode, use the *consistent=y* option of the exp command (as discussed in 6.2.2, “Export/Import” on page 83):

```
exp userid=system/manager full=y consistent=y file=/oracle/orfs.dmp
```

3. Use ADSM to back up (or archive as our example shows) the file containing the exported database:

```
dsmc archive -de=”ORFS FULL EXPORT 1995-09-18” /oracle/orfs.dmp
```

4. Back up the parameter files:

```
dsmc archive -de=”ORFS FULL EXPORT 1995-09-18” ”/oracle/dbs/initorfs*”
dsmc archive -de=”ORFS FULL EXPORT 1995-09-18” ”/oracle/dbs/configorfs*”
```

6.9.2 Restoring the Database

To perform full recovery you must:

1. Restore the parameter files:

```
dsmc retrieve -de=”ORFS FULL EXPORT 1995-09-18” ”/oracle/dbs/initorfs*”
dsmc retrieve -de=”ORFS FULL EXPORT 1995-09-18” ”/oracle/dbs/configorfs*”
```

2. Use ADSM to retrieve the file containing the exported database:

```
dsmc retrieve -de=”ORFS FULL EXPORT 1995-09-18” /oracle/orfs.dmp
```

3. Connect to the database and shut down and start up the database in exclusive mode:

```
sqldba lmode=y
connect internal;
shutdown normal;
startup open exclusive;
exit;
```

4. Import the database.

   From the AIX command line issue this imp command:

```
imp userid=system/manager full=y file=/oracle/orfs.dmp
```

Chapter 7. Oracle7 Enterprise Backup Utility

This chapter describes how to use the Oracle7 Enterprise Backup Utility (EBU) in conjunction with ADSM to back up, restore, and recover Oracle7 databases.

EBU can perform offline and online backups of the database, tablespaces, data files, control files, archive logs, and database parameter files. Recovery can be performed for the database, tablespaces, and individual data files. Recovery can be performed to the databases most current state or to a point in time. EBU provides the interface to the Oracle7 database and the functions for backup, restore, and recovery. It does not, however, provide any storage management capabilities other than the ability to create backups on local disks. EBU must be integrated with other storage management products such as ADSM to provide a complete enterprisewide storage management solution. This integration is enabled through the ADSMConnect Agent for Oracle. The ADSMConnect Agent for Oracle is currently available for the following platforms:

- AIX 4.1 and above
- HP-UX 10.20 or 11.0
- Solaris 2.5.1

At the time of writing IBM plans to introduce an ADSMConnect Agent for Oracle on Windows NT.

The following topics are covered in this chapter:

1. Overview
2. Functions
3. ADSMConnect Agent
4. Installation and configuration
5. Tools and utilities
6. Backup and recovery examples
7. Administration

This chapter is an introduction to EBU and in particular how it integrates with ADSM. It is not an authoritative guide to EBU or Oracle7 backup and recovery. This chapter should be read in conjunction with the *Oracle7 Enterprise Backup Utility Administrator's Guide*.

OBACKUP and EBU 2.2

EBU was previously called OBACKUP and is sometimes still referred to as such. In previous versions of EBU, the primary command to perform backup and restore operation was also **obackup**. With EBU 2.2 this command is now **ebu**. The **obackup** command is still provided for backward compatibility.

With previous versions of EBU there were two separate EBU products: EBU-S for the Oracle7 Server product and EBU-PV for the Oracle7 Parallel Server. With Version 2.2 EBU is now a single product that supports all Oracle7 implementations.

7.1 Overview

EBU is a backup utility provided with Oracle7 that integrates Oracle7 databases into an enterprisewide backup solution. This section presents a brief overview of the EBU architecture, components, and backup catalog.

7.1.1 Architecture

EBU provides backup and restore functions for Oracle7 databases. When invoked, EBU automatically identifies the database objects (data files, control files, and archived logs) and performs a backup or restore operation for the selected database. EBU provides integration with Oracle7 but has no storage media management capabilities. It relies on storage management functions provided by third-party media management vendors. IBM supplies these functions through the ADSMConnect Agent for Oracle.

The ADSMConnect Agent provides function calls that enable EBU to send data objects to and from an ADSM server. These function calls are defined in the Oracle SBT media management API. The ADSMConnect Agent is implemented as a library (libobk.a) that uses standard ADSM API functions to communicate with an ADSM server. During the EBU installation process, the EBU tools are linked with this library so that Oracle7 database objects can be sent to and received from an ADSM server. Figure 43 illustrates how EBU uses the SBT function calls with the ADSMConnect Agent to send backup data to an ADSM server.

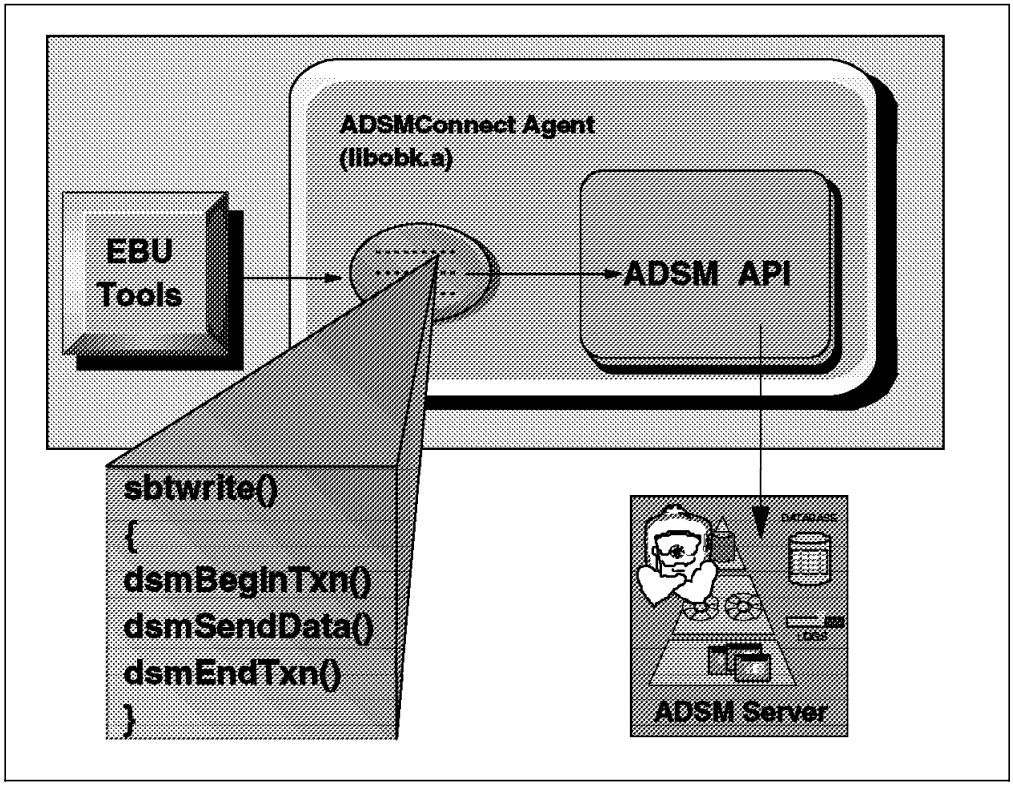


Figure 43. EBU and ADSMConnect Agent Architecture

When EBU backs up a database instance, it sends the backup data to the output media by invoking the *sbtwrite*{} function within the libobk.a library. This function is defined in the SBT API but implemented within the ADSMConnect

Agent, which provides the libobk.a library. The ADSMConnect Agent then uses ADSM API functions to perform the requested operation to the ADSM server. In this example an ADSM transaction is performed to send data from EBU to the ADSM server.

7.1.2 System Components

EBU consists of several components that interact during the backup and restore processes. Figure 44 illustrates these components and the steps involved when EBU backs up a database to an ADSM server through the ADSMConnect Agent.

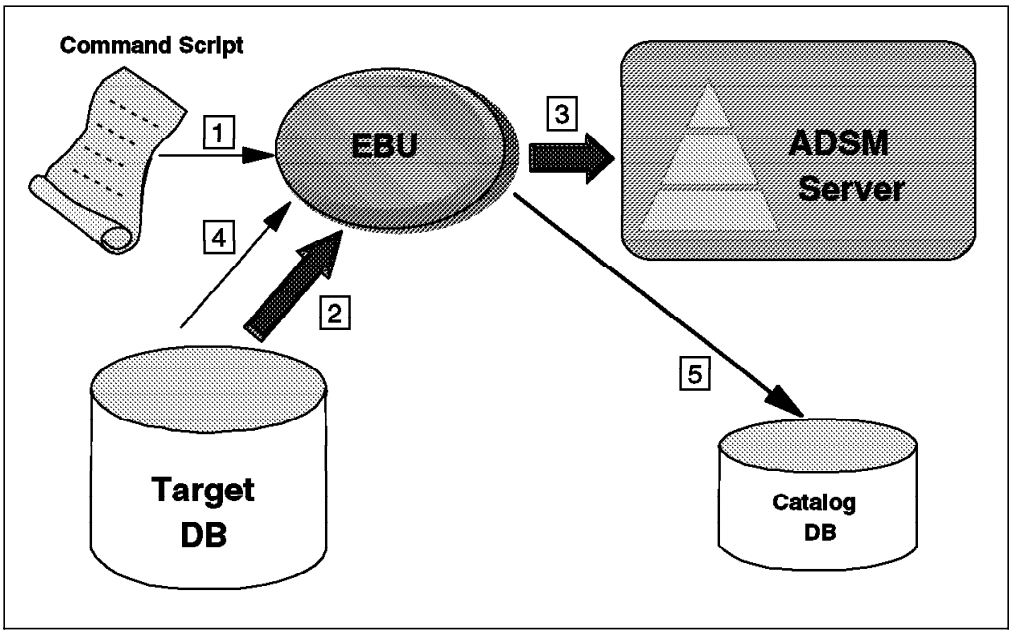


Figure 44. EBU System Components

An Oracle7 database instance that is to be backed up through EBU is called a target database. EBU is invoked by defining a command script that defines the backup or restore function to perform (1). EBU then executes the function (in this example a database backup) and sends the database backup objects (2) by means of the ADSMConnect Agent to the ADSM Server (3).

Although these backup objects are held on the ADSM server, EBU maintains historical backup and configuration information about the target database (4) in a separate Oracle7 database instance called the backup catalog database (5).

7.1.3 Backup Catalog

The backup catalog is a shared repository of EBU information about one or more target databases. The backup catalog is an Oracle7 database, typically separate from the target databases. EBU obtains information from the backup catalog to determine how to execute backup and restore operations. The data stored in the backup catalog comprises structural information about the target databases and historical information about previously performed backup and restore jobs.

Each target database managed with EBU is registered with the backup catalog through EBU commands. During online backups EBU automatically updates the information in the backup catalog to reflect the current structure of the target database. EBU commands can be used to manage the inventory of backups that have been performed on target databases and stored on the ADSM server. For

example, an old backup job can be deleted, which will delete the job record from the backup catalog and inactivate the corresponding backup objects on the ADSM server.

Backup catalog availability is essential to the operation of EBU. EBU automatically backs up the catalog when target databases are backed up. These catalog backups are stored on the ADSM server along with the target database backups. The backup catalog is a shared resource potentially containing information about multiple target databases. Therefore, to reduce the complexity of a disaster recovery affecting one or more of your target databases, it is strongly recommended that the catalog database instance be on a machine that contains no target databases.

7.2 Functions

This section describes the various function of EBU. Topics covered are:

- Database backups
- Performing backups
- Database restores
- Database recovery
- Performing restore and recovery

7.2.1 Database Backups

EBU can back up Oracle7 databases installed on file systems, raw devices, and virtual shared disks (VSDs) that are used with the Oracle Parallel Server (OPS). The following types of backup can be performed:

- Online backup
- Offline backup
- Archived redo log backup
- Control file backup
- EBU catalog backup

7.2.1.1 Online Backup

An online backup can be performed while the target database is online and being accessed by users. To perform online backups the database must be operating in ARCHIVELOG mode (see 6.1, “Oracle DBMS Structure” on page 81) so that online redo logs are archived. During the restore of a database that was backed up online, these redo logs are applied to recover the restored database to a consistent state.

An online backup can back up all tablespaces, control files, parameter files, and archived tablespaces within a database, or just a subset of the tablespaces. A full online backup that includes all database files is performed through the **backup database** command. A partial online backup that includes only a subset of tablespaces is performed through the **backup <subset>** command where <subset> specifies the individual data files or tablespaces within the database to be backed up. These commands are defined in EBU command scripts, which are covered in more detail in 7.5, “Tools and Utilities” on page 122.

7.2.1.2 Offline Backup

An offline backup consists of all tablespaces, control files, parameter files, and archived redo logs. An offline backup can be taken only after the database is completely shut down. Failure to shut down the database will cause an offline backup to fail. An offline backup is also known as a point-in-time backup. If the target database is running in NOARCHIVELOG mode, an offline backup is the only backup function that can be performed.

Before running the offline backup job, EBU starts the database in mount mode so that it can query the current database information against the information stored in the catalog to determine any structural changes. EBU mounts the database, using the default parameters from the initSID.ora file. The initSID.ora file contains instance-specific tuning and configuration parameters used to configure the instance.

7.2.1.3 Archived Redo Log Backup

EBU backs up the archived redo logs by default during both online and offline backups. EBU identifies the archived redo log files by querying the target database for the current LOG_ARCHIVE_DEST location specified by parameters in initSID.ora. EBU then constructs a job that backs up the archive logs to ADSM. The location where EBU searches for the archive logs can be altered by specifying an alternative location in the EBU command script.

EBU, through the **arch_copies** option, also allows saving more than one copy of the archived redo logs. The maximum number is four copies.

By default the archived log files are not deleted after each backup. They can be deleted by using the **archdelete** option in the EBU command script. EBU does not delete any archived log files that have not been backed up. If they are not deleted EBU includes them in all future backup jobs that archive the logs. The decision to delete or not depends on the available disk space and the time available for recovery.

7.2.1.4 Control File Backup

Control files keep track of the database and its physical file structure. You should back up the control files any time you make a structural changes to a database.

EBU backs up the control files whenever a database file or archive log is backed up. EBU obtains the name of the control file from the EBU catalog and backs it up. Control files are backed up after the data files because they store all header information for the data files that is required for recovery. If the control file is mirrored, EBU backs up only one copy of the mirror.

7.2.1.5 EBU Catalog Backup

The EBU catalog is automatically backed up after every backup job that includes at least one database object. This default option can be disabled by using the **catalog=none** option in the backup script. The catalog can also be backed up separately by using the **backup catalog** command with a command script.

If the catalog backup fails, the backup of the target database is still marked successful, but the backup job completes with a warning.

7.2.2 Performing Backups

To perform EBU backups you prepare a backup command script. This script contains commands that determine which backup functions will be performed. Once you have prepared a backup script, execute it through the **ebu** command.

Perform the following steps to execute a backup operation:

1. Verify that the backup catalog database is accessible.
2. Prepare a backup command script.
3. Verify that the target database is in the correct state (open for online backups and closed for offline backups).
4. Invoke the backup job.

   You can invoke the job in one of the following ways:

   - Directly, using the **ebu** command from an AIX shell prompt:

     ```
     % ebu script_name
     ```

   - Embed the **ebu** command in a script with other commands that perform pre- and post-backup activities
   - From ADSM by defining a schedule that executes the **ebu** command or a script containing the command.

7.2.3 Database Restores

A restore operation retrieves backups from the backup media, in these examples, the ADSM server. The following types of restore operations can be performed:

- Database restore
  - Consistent
  - Inconsistent
  - Point in time
- Restore <subset>
- Archived log restore
- Backup catalog restore

7.2.3.1 Consistent Database Restore

A consistent restore brings all data files in the database to a consistent state, and recovery is not needed to open the database. This type of restore can be performed only with an offline backup of the database. EBU uses the latest offline backup for the restore operation.

7.2.3.2 Inconsistent Database Restore

An inconsistent database restore restores all files from the most current backup. The files restored are inconsistent because the backup being restored was performed while the database was open and in use. After the restore the database must be recovered to a consistent state before it can be opened. Before the restore the database must be fully shut down. During the restore the database will be started and mounted exclusively by EBU.

7.2.3.3 Point-in-Time Restore

By default, EBU restores the latest backup of a database or data file. If you want to restore the database or the data files, using a backup older than the latest, you can use a point-in-time restore.

7.2.3.4 Restore <Subset>

This type of backup restores a subset of the database files. Four objects can be restored:

- **Data files:**

  Data files are restored to the same location from which they were backed up. A data file is restored over the original file even if the old data file still exists. If a data file is restored while the database is online, the data file must first be taken offline.

  To restore offline data files that belong to a tablespace, you have to explicitly specify this within the **restore <subset>** command. It is not possible to restore offline data files with **restore database**.

- **Control files**

  You can restore control files to either the original location or another location. To restore control files to the original location, the database must be shut down.

- **Parameter files**

  You can restore parameter files at any time with the database either online or offline. If the parameter file still exists, it is not restored with its original name. It is restored with the same name but with a suffix of *<jobid>* added. Otherwise it is restored to its original location.

- **Archive logs**

  You can restore the archive log files at any time with the database either online or offline. If the archive redo log file does not exist, you can restore it to the original location.

7.2.3.5 Archived Log Restore

Archived redo logs are restored automatically whenever at least one data file is restored. They are not restored by default when you restore control files or parameter files, but you can specify that they be restored.

7.2.3.6 Backup Catalog Restore

If the backup catalog is lost, it can be restored through EBU. It is not necessary to have a recovery catalog available in order to restore it. EBU can determine which is the latest backup of the catalog and uses that information to restore it. By default the **restore catalog** command restores the most current EBU catalog backup and overwrites the existing EBU catalog in the catalog database.

EBU also allows the catalog to be restored to a different database through the **AS** specifier in the restore catalog script. EBU also enables you to restore a backup of the EBU catalog that is not the latest backup. When the catalog is restored to a point in time, any jobs executed on the target databases after that point in time will be out of synchronization with the newly restored backup catalog.

7.2.4 Database Recovery

When an inconsistent restore is performed, the restored data files have to be made consistent with each other and with the control file. This procedure is called recovery. EBU can automatically perform the recovery for most situations. Recovery is required before the database can be opened. If the **recover** option is specified in the restore script, EBU automatically performs the recovery after the restore. If the recovery option is not specified, manual recovery is required. EBU recovery consists of the following steps:

1. If the database is shut down, EBU starts and mounts it.
2. EBU runs Oracle7 recovery, using an **alter database recover** command.
3. EBU returns the database to the state it was in before the recovery was started.

7.2.5 Performing Restore and Recovery

Run the following steps to perform a restore:

1. Verify that you have the necessary permissions to restore or recover a target database.
2. Verify that the backup catalog database is accessible.
3. Prepare a restore command script.
4. For restore <subset>, take the tablespaces or data files to be restored offline; for restore database, shut down the database.
5. Run the restore job.

7.3 ADSMConnect Agent

The ADSMConnect Agent contains the media management library (libobk.a) used by EBU to access external storage management services. Installation of this library enables EBU to send and receive backup objects from ADSM.

This section covers installation and configuration of Versions 2.1.0.6 and 2.1.0.7 of the ADSMConnect Agent on AIX.

7.3.1 Installation

This section covers installation of Versions 2.1.0.6 and 2.1.0.7 of the ADSMConnect Agent for AIX. To install the ADSMConnect Agent, use the System Management Interface Tool (SMIT) or the AIX **installp** command. The following file sets must be installed:

- adsmagent.aob.obj
- adsmagent.aobord.obj

The ADSMConnect Agent installation path is */usr/lpp/adsmagent/aob.*Table 3 lists the files installed. The files differ slightly for the different versions.

Table 3 (Page 1 of 2). ADSMConnect Agent Files

	Description
adsmaob.ord	Version 2.1.0.6 verification file
agent.lic	Version 2.1.0.7 verification file
dsm.opt.smp	Sample client user options file

Table 3 (Page 2 of 2). ADSMConnect Agent Files	
	Description
dsm.sys.smp	sample client system options file
en_US/dsmclient.cat	Version 2.1.0.6 message catalog
en_US/dsmclientv3.cat	Version 2.1.0.7 message catalog
libobk.a	Shared library
aobpswd	Password manager program
options.doc	Documentation for client options
README.AOB	Readme file

One of the main differences between the two levels of the ADSMConnect Agent is the way that the libobk.a library is built. The libobk.a library provided with Version 2.1.0.6 of the ADSMConnect Agent has Version 2.1.0.6 of the ADSM API statically linked within it. Therefore separate installation of the ADSM API client is not necessary. The libobk.a library with Version 2.1.0.7 of the ADSMConnect Agent is different. It is dynamically linked to Version 3.1.3 of the ADSM API client, and Version 3.1.3 or later of the API client must be installed separately (not covered here).

As part of the ADSMConnect Agent installation, a symbolic link is created from */usr/lib/libobk.a* to *libobk.a* in the ADSMConnect Agent library.

EBU Migration Considerations

If EBU is already being used when the ADSMConnect Agent is installed, the backups taken prior to the installation will no longer be accessible. The installation process updates the symbolic link with the new libobk.a version supplied with the ADSMConnect Agent. Backups taken before installation will have been using a different libobk.a version to store backup data. This previous version of the library is no longer linked, and as a consequence the old backups are not accessible. Care should be taken to maintain a copy of any previous libobk.a versions that may be required for recovery purposes. To use these previous libobk.a versions again, the symbolic link from /usr/lib/libobk.a must be changed to point to the required version of libobk.a.

7.3.2 Configuration

After the ADSMConnect Agent is installed, it must be configured. Configuration involves updating the dsm.sys and dsm.opt files, registering the nodename on the ADSM server, defining environment variables, initializing the ADSMConnect Agent password, and defining a management class on the ADSM server for Oracle backups.

7.3.2.1 Edit Options Files

The ADSMConnect Agent uses client options files, dsm.sys and dsm.opt, similar to those of the AIX backup-archive client. Installation of the ADSMConnect Agent creates samples of these files (Table 3 on page 112) in the */usr/lpp/adsmagent/aob* directory. A dsm.sys options file for the ADSMConnect Agent must be created in this directory with the correct ADSM server and communication method options (Figure 45 on page 114).

```
SErvername  yellow
   COMMmethod         TCPip
   TCPPort            1500
   TCPServeraddress   yellow

   PASSWORDACCESS     prompt

   INCLEXCL           /usr/lpp/adsm/bin/inclexcl
```

Figure 45. Sample dsm.sys for ADSMConnect Agent

Ensure that the PASSWORDACCESS option is not set to generate in the dsm.sys file (prompt is the default if not specified). If it is set to generate, EBU will not work because the generate function forks a separate process, which conflicts with Oracle's message-passing activities. Using prompt does not fork a process.

7.3.2.2 Register Nodename

The nodename for the ADSMConnect Agent must be registered on the ADSM Server through the BACKDELETE=yes option. The name must match the nodename specified in the ADSMConnect Agent dsm.sys file if specified. If a nodename is not specified in dsm.sys, it defaults to the host name of the system.

7.3.2.3 Initialize Password

The ADSMConnect Agent provides a program, aobpswd, to simulate PASSWORDACCESS=GENERATE. The program eliminates the need to manually enter a password, allowing EBU to contact the ADSM server without prompting each time for a password. The aobpswd program encrypts the ADSM password and stores it in a secure file located with the ADSM DSMO_PSWDPATH environment variable. The password is then passed by the ADSM API to the ADSM server without user intervention.

Note

A key difference between the aobpswd program and passwordaccess generate is that the aobpswd program does not automatically update the password when it expires, that is, it does not automatically regenerate the password. The backup jobs fail when the passwords are no longer valid. Password changes should be scheduled before expiration has occurred.

Before you use the ADSMConnect Agent, aobpswd must be run as the root user.

To initialize the ADSMConnect Agent password, perform the following steps:

1. Start the aobpswd program to set up the connection with the server that you specified in dsm.opt.
2. Enter your current node password. You are prompted for a new password. You do not have to enter any value; just press Enter. The following message is displayed:

```
Your password has been written to the File.
Verify that the DBA has read access to the
ADSMO.yourhostname file.
```

Make sure that the user who will use EBU has read permission to this file.

You can use aobpswd to update the password when it expires.

7.3.2.4 Define Environment Variables

The ADSM API uses unique environment variables to locate files. Thus for API applications it can use different files from those used by the backup-archive client. These environment variables must be defined in the user profile (.kshrc, .cshrc, or .profile) of the user account that will run EBU backup and recovery jobs. The syntax for defining environment variables is:

Korn shell (ksh): export VAR_NAME=value

```
export DSMI_CONFIG=/home/yourlogin/dsm.opt
```

C shell (csh): setenv VAR_NAME value

```
setenv DSMI_CONFIG /home/yourlogin/dsm.opt
```

The following environment variables are required for the ADSMConnect Agent:

DSMI_CONFIG Fully qualified name that points to the dsm.opt file.

DSMI_DIR Fully qualified name that points to the directory that contains the dsm.sys file and the verification file.

DSMI_LOG Fully qualified name that points to the directory that contains the API error log file (dsierror.log).

The following environment variables are optional:

DSMO_AVG_SIZE The average size of objects. This variable is passed to the ADSM server where it is used to determine storage pool and device usage. It is recommended that the DSMO_AVG_SIZE environment variable be set. In Version 2.1.0.6, the units are megabytes, with a default of 50 MB.

During normal ADSM client to ADSM server backup operations, the ADSM client notifies the ADSM server of the size of the next object that will be sent. The ADSM server determines whether space is available to store the object before allowing the object to be sent. With EBU the server is not notified of the object size; rather it is given the value determined by the DSMO_AVG_SIZE environment variable. If this value is set lower than the actual object size, the ADSM server must halt transfer operations to determine whether additional space is available. This process can cause problems on the ADSM server. It is recommended that the DSMO_AVG_SIZE environment variable be set to a number equal to the size of the largest Oracle data file within the database. When you set the size of the variable, pay attention to threshold maximums set on ADSM server storage pools. For example, if a disk storage pool accepts files to a maximum of 20 MB and the value of DSMO_AVG_SIZE is set to 40000 (40 MB), the object would be forced to the next storage pool (usually tape).

DSM_FS The filespace name that will be created on the ADSM server. This environment variable gives you control over the filespace name for the EBU backup. In fact, you can use

several different names to differentiate your backups. When setting up this option, do not use a / before the filesystem name. The default is /adsmorc.

Note

If you use many different filespace names, you must keep track of which name you use for each backup and ensure that you use the same name on the environment variable when you issue a restore.

DSMO_NODE The node name containing a string of 1 to 64 characters. This environment variable is used to specify a unique node name for the ADSMConnect Agent.

The default is the value returned by the AIX **hostname** command or the **NODENAME** value in the dsm.opt file.

DSMO_OWNER The session and backup object owner name containing a string of up to 64 characters. The default is a value returned by the **getuid** command. Use of this environment variable is not recommended.

DSMO_PSWDPATH This environment variable is pointing to the directory where the **ADSMO.yourhostname** password file resides. The default is the current working directory.

7.3.2.5 Initialize Password

The ADSMConnect Agent provides a program, **aobpswd**, to simulate PASSWORDACCESS=GENERATE. The program eliminates the need to manually enter a password, allowing EBU to contact the ADSM server without prompting each time for a password. The aobpswd program encrypts the ADSM password and stores it in a secure file located with the ADSM **DSMO_PSWDPATH** environment variable. The password is then passed by the ADSM API to the ADSM server without user intervention.

Note

A key difference between the aobpswd program and passwordaccess generate is that the aobpswd program does not automatically update the password when it expires, that is, it does not automatically regenerate the password. The backup jobs will fail when the passwords are no longer valid. Password changes should be scheduled before expiration has occurred.

Before using the ADSMConnect Agent, you must be run as the root user.

To initialize the ADSMConnect Agent password, follow these steps:

1. Start the aobpswd program to set up the connection with the server that you specified in dsm.opt.
2. Enter your current node password. You are prompted for a new password. You do not have to enter any value; just press Enter. The following message is displayed:

```
Your password has been written to the File.
Verify that the DBA has read access to the
ADSMO.yourhostname file.
```

Make sure that the user who will use EBU has read permission to this file.

You can use aobpswd to update the password when it expires.

7.3.2.6 Define Management Class

As both backup-archive and API clients can be used on the same node, it is recommended that separate management classes be used for the different file systems.

EBU manages it own inventory of backups, using the ADSMConnect Agent and the ADSM server as a secure repository to maintain them. EBU determines the names of the individual objects that are stored on ADSM. Each object has a unique name, so each successive backup of the same data has a different name on the ADSM server. Therefore only one copy of an object from EBU is ever stored on the ADSM server. All EBU objects stored on the ADSM server remain active (because they are the only copy) and can be individually deleted only when EBU issues a delete object API call. When EBU has deleted the object, it can be discarded on the ADSM server.

A separate management class should be created with a backup copy group (the ADSMConnect Agent does not use archiving) defined that maintains only one active version of the backup object and has the following parameters for deleting expired backup objects:

- VERDELETE=0
- RETONLY=0

With such a management class only one copy of the backup object will be held on the ADSM server. When EBU deletes the object from the backup catalog and on the ADSM server, it issues a delete object API call to change the status of the object from active to inactive on the ADSM server. This will be expired the next time that inventory expiration is run.

When a database is backed up, the default management class for the node name is used unless you override it with a different value in an include-exclude file by adding to dsm.sys an INCLECXL option that points to the client include/exclude list. The following sample statement in the include-exclude file binds all backups belonging to the ADSMConnect Agent filespace, /adsmorc, to the ORACLE management class:

```
INCLUDE /adsmorc/.../* ORACLE
```

7.4 Installation and Configuration

Before EBU can be installed and configured, the Oracle environment for the target and backup catalog database instances must be set up.

The following Oracle products are required:

- ORACLE Server Version 7.1.6 or higher
- SQL*DBA or Server Manager
- SQL*Net Version 2.
- PL/SQL Version 2.3.4
- SQL/Plus Version 3.3.4 or higher

For this project EBU Version 2.2 was installed on AIX 4.1.4 with Oracle Version 7.3.4.

SQL*DBA

SQL*DBA is no longer supported on Oracle Version 7.3 and up. Server Manager is used to execute SQL commands and replaces SQL*DBA. SQL Server Manager runs on Oracle Version 7.0.0 and higher.

Also required is what Oracle calls third-party media management software, which in these examples is ADSM. EBU should be installed only after the ADSMConnect Agent is installed. Before proceeding with EBU installation, the following must be done:

1. Install backup catalog
2. Configure listener process

7.4.1 Install Backup Catalog

The EBU catalog is a collection of tables in an Oracle7 database. It contains structural information about the target database as well as backup and restore history for each target database. The EBU catalog database is separate from the target databases.

It is recommended that the backup catalog be installed on a separate machine from all target databases and dedicated for use by the backup catalog.

To install an Oracle7 database you use the Oracle installation utility, orainst. This utility is covered in the *Oracle7 Installation Guide* and is not explained further here.

Once an Oracle7 instance for the backup catalog is installed, a separate tablespace can be created for the backup catalog information. By default the backup catalog information is created in the system tablespace. It is recommended that a separate tablespace be created for the backup catalog, especially if the catalog is on an existing database.

The following example creates a tablespace named **ebu** that contains a single 20 MB data file named **ebu01.dbf**:

```
% svrmgrl system/manager@CAT7
svrmgrl> create tablespace ebu datafile 'ebu01.dbf' size 20M;
```

This tablespace will be used when the target database is registered to this backup catalog.

7.4.2 Oracle Listener Process

The listener process enables communication between all of the databases in an Oracle networked environment. It must be configured on all Oracle7 instances that will communicate. The following steps should be performed:

1. When Oracle7 is installed it creates an Oracle user account. This account is used for administration purposes. Switch to the Oracle user and ensure that these environment variables are set:

- ORACLE_HOME
- ORACLE_SID
- ORACLE_BASE
- ORACLE_DOC

2. As the Oracle user, change to the $ORACLE_HOME/network/admin directory on the target database and ensure that the listener process is correctly configured by checking the listener configuration file, listener.ora. The HOST and PORT should be defined in the LISTENER stanza, and the database instances being listened for should be defined in the SID_LIST_LISTENER stanza:

```
% cat $ORACLE_HOME/network/admin/listener.ora

LISTENER =                    ◄
 (ADDRESS_LIST =
   (ADDRESS =
    (PROTOCOL = TCP)
    (HOST = bering)           ◄
    (PORT = 1521)             ◄
   )
 )

SID_LIST_LISTENER =           ◄
 (SID_LIST =
   (SID_DESC =
    (SID_NAME = DFM7)         ◄
    (ORACLE_HOME = /oracle/app/oracle/product/734 )
    (GLOBAL_NAME = DFM7)
   )
 )
```

   Check the same definitions on the Oracle7 instance being used for the backup catalog.

3. Ensure that $ORACLE_HOME/network/admin/tnsnames.ora on both the target and backup catalog databases has a connect string for each of the database instances that will be used:

```
% cat $ORACLE_HOME/network/admin/tnsnames.ora
  ⋮
     DFM7 =                           ◄
       (DESCRIPTION =
         (ADDRESS = (PROTOCOL = TCP)(HOST = bering)(PORT = 1521))
         (CONNECT_DATA =(SID = DFM7))
       )


     CAT7 =                           ◄
       (DESCRIPTION =
         (ADDRESS = (PROTOCOL = TCP)(HOST = black)(PORT = 1521))
         (CONNECT_DATA =(SID = CAT7))
       )
  ⋮
```

   The names of the connect strings (DFM7, CAT7) are the user definable aliases. The tnsnames.ora file must contain the connect strings for all of the databases you use. The PROTOCOL, PORT, HOST, and SID values depend on your environment. In our environment we used PROTOCOL TCP and PORT 1521.

4. Start the listener process.

Having checked that each database has its listener instance and connect string defined, start the listener process. The **lsnrctl** command can be used to start, check the status of, and stop the listener process.

Starting the listener process:

```
%lsnrctl start
   ⋮
Starting /u01/app/oracle/product/734/bin/tnslsnr: please wait...
   ⋮
  DFM7          has 1 service handler(s)
The command completed successfully
```

Checking the listener process status:

```
%lsnrctl status

   ⋮
Connecting to (ADDRESS=(PROTOCOL=TCP)(HOST=bering)(PORT=1521))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
   ⋮
Listener Parameter File   //u01/app/oracle/product/734/network/admin/listener.ora
Listener Log File         /u01/app/oracle/product/734/network/log/listener.log
Services Summary...
  DFM7          has 1 service handler(s)
The command completed successfully
```

Stopping the listener process:

```
% lsnrctl stop
%lsnrctl stop
   ⋮
(c) Copyright 1997 Oracle Corporation.  All rights reserved.

Connecting to (ADDRESS=(PROTOCOL=TCP)(HOST=bering)(PORT=1521))
The command completed successfully
```

Before continuing, make sure that the backup catalog can be accessed from each target database. The **tnsping** command can be used to confirm this:

```
% tnsping CAT7
   ⋮
Attempting to contact (ADDRESS=(PROTOCOL=TCP)(HOST=black)(PORT=1521))
OK (90 msec)
%
```

EBU can now be installed.

7.4.3 Installation

EBU is installed using the Oracle installer program. Before installation make sure that these variables are set correctly:

- $ORACLE_HOME
- $ORACLE_SID
- $ORACLE_PATH
- $ORACLE_DOC

Follow these steps:

1. Mount the Oracle7 installation CDROM.

   The installation of EBU must be performed from the /olink directory, which is linked to the CDROM during the installation. Execute the following commands as the root user to create the /olink directory and mount the Oracle7 installation CDROM:

```
# mkdir /cdrom
# mkdir /olink
# chmod 777 /olink
# mount -rv cdrfs /dev/cd0 /cdrom
# exit
```

2. Run the start.sh script.

   The start.sh startup script creates a link from the /olink directory to the CDROM. Switch to the Oracle user and execute start.sh script in the orainst directory:

```
# su - oracle
% cd /cdrom/orainst
% ./start.sh
```

   When prompted, enter the name of the Oracle link directory: /olink. This script sets up the correct environment within the /olink directory.

3. Run the rootpre.sh script.

   Switch back to the root user and run the rootpre.sh script. This step ensures that the necessary kernel extensions have been loaded and the streams environment has been configured correctly. This script must be run from the olink directory as the root user:

```
% su root
# cd /olink/orainst
# ./rootpre.sh
%
```

   This script loads the necessary AIX kernel extensions for Oracle. After the script has been run, it is recommended that the system be rebooted.

4. Run the Oracle installer.

   As the Oracle user, change to the orainst directory and run the orainst installer program:

```
% cd /olink/orainst
% ./orainst
```

   The Oracle installation dialog will display. From the main screen (not shown) select the **Install New Product - Do Not Create DB Objects** option and follow the installation prompts. The Software Asset Manager Screen (Figure 46 on page 122) will display. Select the **Oracle7 Enterprise Backup Utility** option and **Install**.

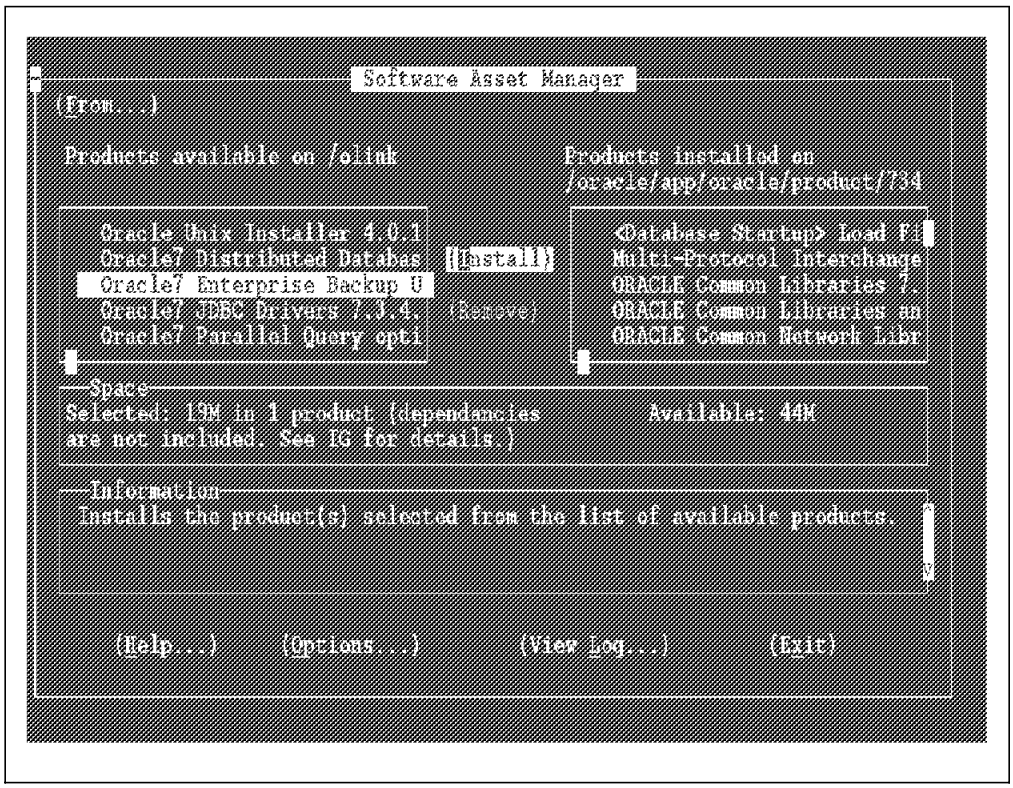


Figure 46. Software Asset Manager EBU Installation Dialog

The EBU installation continues.

5. Run the root.sh script

   After the installation completes, run the root.sh postinstallation script to set the necessary file permissions for Oracle products and perform any postinstall setup activities:

```
% su root
# cd /$ORACLE_HOME/orainst
# ./root.sh
# exit
```

After the installation is complete the environment profiles for the users that will use EBU must be updated. Ensure that the **$EBU_HOME** environment variable is set to the directory where EBU was installed. The default installation directory is $ORACLE_HOME/obackup. The PATH must also be updated to include the $EBU_HOME/bin directory path.

7.5 Tools and Utilities

This section introduces the tools and utilities provided by EBU to help manage and backup Oracle7 databases. The **ebu** command, EBU commands scripts, and **ebutool** command are covered.

The **ebu** command is used in conjunction with EBU command scripts to perform Oracle7 backup and restore tasks. The **ebutool** command is the administrators interface to the backup catalog and the ADSM server.

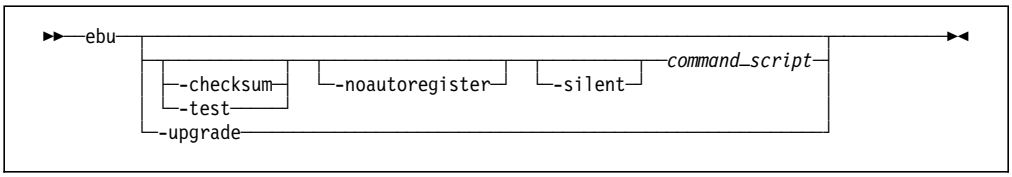
7.5.1 EBU Command

The **ebu** command is used to process an EBU command script on the target database. It enables you to perform the following tasks:

- Register target databases with the backup catalog
- Back up databases, tablespaces, control files, and archive logs
- Restore databases, tablespaces, control files, and archive logs
- Back up and restore the backup catalog

An EBU command script is a plain text file residing in the local filesystem containing command stanzas that specify which functions are performed.

The **ebu** command requires several environment variables to be set: $ORACLE_SID, $ORACLE_HOME, and $EBU_HOME. The $ORACLE_SID variable is set to the Oracle system ID of the instance to be backed up. $ORACLE_HOME and $EBU_HOME point to their respective code directories specified during the installation of these products. Typically the **ebu** command is invoked through an Oracle userid, so these variables should already have been set in the Oracle user's login profile.



The *command_script* parameter defines the name of the EBU command script that EBU will process. In addition the following options can also be specified:

Option	Description
checksum	Enables **ebu** to perform checksums on backup and restore. The checksum is stored in the catalog database to be used in subsequent operations.
test	The test option allows you to test the command script for errors and does not result in any operation to the target database, backup catalog, or ADSM server.
noautoregister	The noautoregister option prevents the automatic update and registration of the target database in the catalog database. Online backups normally perform registration update while offline does not. Registration is required whenever the structure of the target database has changed.
silent	The silent option suppresses progress messages being sent to stdout. However, syntax errors will still be sent to stderr. This option requires a logfile to be specified in the command script.
upgrade	Upgrade is used to upgrade existing catalog databases prior to EBU 2.2.

7.5.2 EBU Command Scripts

EBU uses command scripts to define the operations that are to be performed by the **ebu** command. To use a command script the required and optional stanzas are defined in a plain text file. The **ebu** command is then executed with the command script as an argument:

```
% ebu  command_script
```

Some of the common EBU command script stanzas are:

Stanza	Purpose
backup	Used to perform all backups of target database. Types include *online* and *offline* and scope includes *database* and *tablespace*. Clauses include how to handle the archive logs and to perform catalog backup.
restore	Used to restore all or part of the database and may be used to restore configuration files and the catalog database
include_dbfile	Used to include additional offline data files during an offline backup. Note that checking is not performed on the content of the files specified with this parameter.
recover	Used to perform automatic recovery after the requested object has been restored. This option is not valid with restore consistent.
register	Used to register the target database with the catalog database. The target database must be online to use this command. The *catalog.ebu* configuration file is used to provide the connect string. The command is also used to update the catalog entry for the target database, should its structure change.
log	Specifies the name of the file used to store the messages generated while the **ebu** command is running
trace	Specifies the name of the file used to store trace information useful in debugging problems
mux	Used to specify a list of data files to multiplex to a single backup file set (BFS). It can be used to match the speed of the disk subsystem to capabilities of the backup stream.
parallel	Used to specify the number of concurrent streams to the ADSM server. The number of concurrent streams depends on the configuration of the ADSM server and the network capabilities.

Figure 47 on page 125 illustrates a sample script for backing up the system tablespace online. Note also the additional parameters to the ***backup*** stanza. The layout of this file is free format, and you can arrange the content as you desire with comments preceded by the character #.

```
#EBU command script to backup system table space.
backup online tablespace = "system"
parallel = 1
log = "/u01/app/product/734/obackup/log/ebu@.log"
```

Figure 47. Sample EBU Backup Tablespace Command Script

Special Characters

You can use the special characters ? and @ when specifying data files. They represent $ORACLE_HOME and $ORACLE_SID, respectively.

Additional parameters that affect performance may be included with the **Parameter** option on the command line or in the EBU script. The **Parameter** stanzas takes a file name as an argument with the file containing the following options:

Parameter	**Option**
buff_size	Sets memory buffer size used for I/O transfer
tape_io_size	Used to set the size of each I/O request to backup medium
disk_io_size	Used to set the size of each I/O request to disk

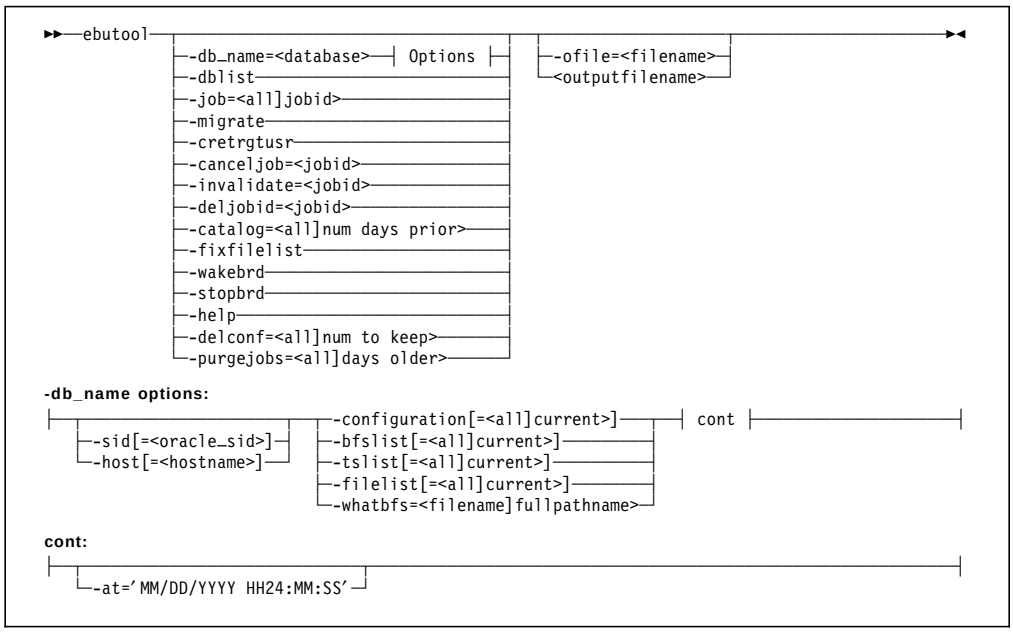
An offline database backup will check to see whether the instance is offline and will abort with error if it is online. If, during the offline backup, the instance is put online by an administrator, the backup will fail due to the instance being open. EBU will delete the BFSs from the ADSM server and update the catalog with a job outcome of "failed." The Oracle instance is shut down by EBU at completion of the failed backup job, and any application or user that may have connected to Oracle will be disconnected.

7.5.3 ebutool Command

The **ebutool** command is used to administer backup operations and to manage backup information held in the backup catalog database. Using **ebutool** you can perform the following tasks:

- Query backup job information and history
- Delete redundant backup job information
- Cancel active backup jobs
- Invalidate backup jobs

The syntax of **ebutool** is:



Option	Description
db_name	Specifies the name of the target database to be selected from available database backups in the backup catalog database. This option has a number of additional options, described below.
dblist	Returns a list of databases in the ebu catalog. It is useful for identifying the structure before running more detailed reports.
job	Returns a detailed report including date, time, job outcome, BFS name, and tablespace information
migrate	Migrates older EBU catalog databases to EBU Version 2.2 catalog scheme
cretrgtusr	Used to create the EBU user in the target database. If the Oracle user does not exist, it is created by EBU.
canceljob	Aborts an active backup job. The *jobid* can be obtained from the job log or the terminal where the **ebu** command was invoked.
invalidatejob	Used to invalidate a particular backup job. This option is required when a problem has occurred at the ADSM server that results in data loss that cannot be recovered. An example would be a corrupt tape with no copygroup backup available. All BFSs for the particular job are removed, and the *jobid* catalog entry is updated to a job outcome of “invalid.”
deljobid	Used to permanently remove a *jobid* from the backup catalog database and release the storage on the ADSM server
catalog	Returns a list of available EBU catalog database backups

fixfilelist	Used to alter the filesystem names to a format suitable for files to be restored on a different operating system platform with different directory naming conventions.
wakebrd	Tests to see whether a BRD process is running. The BRD process is the instance manager started by EBU to clean up after abnormal termination.
stopbrd	Used to stop the BRD process
help	Lists syntax diagram
delconf	Removes old configurations from the EBU catalog. Associated BFSs are also removed from the ADSM server.
purgejobs	Used to delete jobs from the catalog and ADSM server selected by database and optionally older than a number of days

The following additional options are available for the db_name option.

db_name Option	**Description**
sid	The sid qualifier is used to select a particular instance, should there be databases with the same name stored in the catalog
host	Used to further qualify the database and instance, should there be multiple databases and instances stored in the catalog with the same dbname and sid
configuration	Obtains a list of the current registered configuration for the database. The current configuration is the default. All configurations can be specified by using the *all* option.
bfslist	Returns a list of the current BFSs available for the selected database
tslist	Returns a list of tablespace backups available for the selected database
filelist	Used to return the current file configuration and with each file the BFS and *jobid*. Any file that does not have a backup will have a message ″No Backup Job Found.″ If the BFS cannot be found on the ADSM server, the message will request you to invalidate the *jobid*.
whatbfs	Used to determine which BFS contains the requested file for a given database
at	Qualifier for the -db_name options to provide information for the specified date and time

Querying the backup job will most likely be your first task after the EBU installation is complete and you have taken your first backup. The backup job log will indicate both the database backup and catalog backup job numbers, which you can use with the **ebutool** command query functions. You can also substitute the job numbers with the keyword *all* to select all known jobs available in the EBU catalog.

An EBU task can be interrupted by using the *canceljob* option with the *jobid* to be canceled. The *jobid* must be obtained from the EBU job log or stdout from the

screen where the command was started. The job number is not available until the job has finished and the catalog has been updated.

The backup jobs can be deleted or invalidated from the EBU catalog. Deleting removes the catalog and recover space on the ADSM server and storage pools once inventory expiration has completed. Invalidating, however, marks the backup job as invalid, releases ADSM server storage space, but does not remove the catalog entries. The invalidated job is marked invalid and therefore is available only for historical data reference. Deletion would be used, for example, to clean up after a failed backup job had ended. Invalidation would be used, for example, to indicate a record of backup but that the BFSs are no longer held on the ADSM server.

7.6 Backup and Recovery Examples

This section shows some typical backup and recovery examples of using EBU with ADSM:

- Target database registration
- Offline database backup
- Online tablespace backup
- Multiplexed online database backup
- Recovering a data file
- Recovering a database

Each example illustrates the EBU command script used and the output from the job. The following Oracle environment was used for the examples:

- Target database (Oracle 7.3.4) - DFM7
- Catalog database (Oracle 7.3.4) - CAT7

7.6.1 Target Database Registration

This example shows how to use **ebutool** to register a target database with the backup catalog. First a backup user is defined on the target database. This is the user that will perform the EBU operations on the target database. Second, the target database is registered to the backup catalog.

7.6.1.1 Create Target Database Backup User

Each target database must have a backup user, that is, an Oracle user that performs the EBU operations. Creating this user and using it for all EBU operations ensures that EBU has sufficient privileges to execute the commands on the target database. The user is defined by running an **ebutool -cretrguser** command on the target database (Figure 48 on page 129).

```
oracle@bering % ebutool -cretrgtusr
Oracle7 Enterprise Backup Utility Tool: Release 2.2.0.5.0
- Production on Mon Jan 26 14:29:08 1998

Copyright (c) Oracle Corporation 1979, 1996.  All rights reserved.


Please provide EBU User for the "" database [ebudba] =>
Please provide the password for user "SYS" in database "" [change_on_install] =>
Please provide Password for EBU user "ebudba" in target database "" [ebudba] =>

Connecting to target database as user "SYS"
Issuing Statement: CONNECT sys/******* AS SYSDBA;
Connected.
   ⋮
   ⋮
Created Database User "ebudba" in target database
SYSDBA privilege has been successfully granted to ebudba.
oracle@bering %
```

Figure 48. Creating EBU Backup User with ebutool

The utility prompts for the user name on the target database (the default is **ebudba**), the SYS user password, and the password for the user being created. This command can be used again to change any information about the target database backup user.

7.6.1.2 Register Target Database

Before performing a backup or restore, you have to register the target database in the EBU backup catalog. The registration stores essential structural information about the database. Once the database is registered, EBU maintains and updates the configuration information automatically during online backups and verifies that it is up-to-date for offline backup.

To register a target database or upgrade the configuration information, you have to create an EBU command script containing a **register** command. The register command registers the target database with the backup catalog and updates the configuration information for the target database in the backup catalog. Figure 49 illustrates a sample script for registering a target database.

```
# script name: regDFM7
#
# Task: Register DFM7 in the backup catalog
#
register
 db_name=DFM7
 pfile=?/dbs/init@.ora
 log=./register.log
```

Figure 49. Sample EBU Script for Registering Target Database

Note

The @ character is used for ORACLE_SID, and the ? character for ORACLE_HOME. You must use the full path names in the scripts and not use other wildcards or environment variables.

The script is executed by using the **ebu** command with the script name as an option:

```
% ebu regDFM7
```

A dialog is invoked. The first prompt is for the user name, password, and connect string for the backup catalog:

```
Starting Catalog Database verification

Please Enter the Connect information for the catalog database
The user name you enter will be used to store the EBU catalog tables
Username: cat7dba
Password:
Connect String: cat7
```

This user name will be used by EBU to access the backup catalog. Any user name can be entered. If a user name is entered that does not already exist, EBU prompts for the Oracle7 SYS user password so that it can be created:

```
EBU has detected that you entered an invalid username or password
Do you wish to re-enter the username/password (R to re-enter)
or have EBU create the user (or reset the password) (C to Continue)
or abort (A)?
Enter C to continue, R to re-enter, or A to abort: c

Please Enter SYS password to the catalog database
This is needed to create the "cat7dba" database account you just entered
Password:
```

The next prompt is for the tablespace name on the backup catalog database that will be used for the backup catalog information (defined in 7.4.1, “Install Backup Catalog” on page 118):

```
Please enter the tablespace name for the catalog tables ·SYSTEM“: ebu
  Issuing Statement: @@@ownercre cat7dba <pwd> cat7 ebu

 Starting catalog creation
    Issuing Statement: @@@catcre6;
    Catalog successfully created.
  Ended catalog creation
  Catalog Database verified
Ending Catalog Database verification
```

The necessary tables in the backup catalog are created. The next prompt is for the target database backup user and the connect string:

```
Please input information to login to Target Database
An empty return indicates to use user internal with default local connection
Username: ebudba
Password:
Connect String: dfm7
```

After the values have been entered, the target database is registered. Figure 50 on page 131 shows the log output from the register script job.

```
Oracle7 Enterprise Backup Utility: Release 2.2.0.5.0 - Production on Tue Feb  3 12:12:13 1998

Copyright (c) Oracle Corporation 1979, 1996.  All rights reserved.

CORE Version 3.5.4.0.0 - Production
NLSRTL Version 3.2.4.0.0 - Production

ORACLE_HOME = /u01/app/oracle/product/734                  1
System name:      AIX
Node name:        bering
Release:  1
Version:  4
Machine:  000027463400
Tape Management Software: Release 2.1.6.0
Tape API: Release 0.1
Starting parsing of command script "regDFM7"               2
      #EBU script to register instance DFM7. (2/2/98)
      #
      #
      #
      register
      db_name=DFM7
      pfile=?/dbs/init@.ora
      log=./register.log

Ending parsing of command script

Starting Catalog Database verification             3
  Issuing Statement: @@@ownercre cat7dba <pwd> cat7 ebu

  Starting catalog creation
    Issuing Statement: @@@catcre6;
    Catalog successfully created.
  Ended catalog creation
  Catalog Database verified
Ending Catalog Database verification

Starting Instance Manager verification
  Found Instance Manager (brd) process running with pid = 17848
Ending Instance Manager verification


Starting Target Database state verification        4
  Connecting to Target Database
        ORACLE_SID: DFM7
      ORACLE_HOME: /u01/app/oracle/product/734
           DBNAME: DFM7
         Username: ebudba
    Connect String: DFM7

  Target database state OK
Ending Target Database state verification


Register job 1 started on 04-FEB-98 11:06          5

  Starting Database configuration verification

    Target Database information not found in Catalog.
    Target Database is online - proceeding to register.



  Starting Database configuration verification

    Files registered in Catalog are:               6
      Tablespace "SYSTEM"
        Datafile "/u01/oradata/DFM7/system01.dbf"
      Tablespace "RBS"
        Datafile "/u01/oradata/DFM7/rbs01.dbf"
      Tablespace "TEMP"
        Datafile "/u01/oradata/DFM7/temp01.dbf"
```

Figure 50 (Part 1 of 2). Sample EBU Register Database Job Output

```
      Tablespace "TOOLS"
        Datafile "/u01/oradata/DFM7/tools01.dbf"
      Tablespace "USERS"
        Datafile "/u01/oradata/DFM7/users01.dbf"
      Tablespace "TESTMUX"
        Datafile "/u01/oradata/DFM7/test1.dbf"
        Datafile "/u02/oradata/DFM7/test2.dbf"
        Datafile "/u03/oradata/DFM7/test3.dbf"
      Control file "/u01/oradata/DFM7/control01.ctl"
      Control file "/u01/oradata/DFM7/control02.ctl"
      Control file "/u01/oradata/DFM7/control03.ctl"

      Parameter file "/u01/app/oracle/product/734/dbs/initDFM7.ora"

    Target Database information found to be out of date in catalog
    Automatic catalog update will take place

    Additional Database information:
        CREATION DATE: 03-FEB-98 15:04
               DOMAIN: WORLD

    Registering Target Database in Catalog Database
    Target Database registered in Catalog Database on 04-FEB-98 11:21

  Ending Database configuration verification

Register job 1 SUCCESSFUL on 04-FEB-98 11:06
```

Figure 50 (Part 2 of 2). Sample EBU Register Database Job Output

From the log the various stages of the register job can be observed:

1 EBU identifies the working environment including the Tape Management Software version, which in our case is the ADSMConnect Agent Version 2.1.6.

2 The command script is parsed and written to the log.

3 The backup catalog tables are created as this is the first target database registered with the backup catalog.

4 Access to the target database is verified.

5 The register job is started with a unique job number.

6 The target database data files to be registered are listed.

This register job should be reissued whenever the target database structure is changed, such as by adding or dropping a tablespace.

Once the target database has been successfully registered with the backup catalog, backups can be performed.

7.6.2 Offline Database Backup

This example shows how to perform a full offline backup of the target database with multiple data streams to the ADSM server.

Note

This method of backup is the only one available if the database is operating in NOARCHIVELOG mode.

This example shows how to perform full offline backup without backing up the archive logs and using two parallel I/O streams from EBU to the ADSM server. Figure 51 on page 133 is a sample script called offl2DFM7 that performs this backup.

```
# script name: offl2DFM7
#
# Task:
# EBU script to perform offline database backup instance DFM7.
# Using two Parallel I/O streams and no archive logs.
#
#
#
backup offline database
 db_name=DFM7
 parallel=2                  1
 archivelog=none             2
 log=./offline.log
```

Figure 51. Sample EBU Script for Full Offline Backup

1 Two parallel I/O streams will be used by EBU, resulting in two parallel sessions to the ADSM server.

2 The archive logs are not backed up at the end of the database backup. The backup process shuts down the database before performing the backup. As a result, the database is static with no updates being made during the backup. Therefore the archive logs are not required for recovery purposes.

The database must be offline before the backup is performed. For this example it was shut down by using the Oracle server manager:

```
SVRMGR> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SVRMGR>
```

Once the database is shut down, the backup script is executed by using the **ebu** command with the script name as an option:

```
% ebu offl2DFM7
```

Figure 52 shows the log file resulting from this job.

```
Oracle7 Enterprise Backup Utility: Release 2.2.0.5.0 - Production on Tue Feb  7 18:13:41 1998

Copyright (c) Oracle Corporation 1979, 1996.  All rights reserved.

CORE Version 3.5.4.0.0 - Production
NLSRTL Version 3.2.4.0.0 - Production


ORACLE_HOME = /u01/app/oracle/product/734
System name: AIX
Node name: bering
Release: 1
Version: 4
Machine: 000027463400
```

Figure 52 (Part 1 of 4). Sample EBU Offline Database Backup Log

```
Tape Management Software: Release 2.1.6.0
Tape API: Release 0.1


BACKUP job started with pid=21336 on 02/07/1998 18:13:41

  Starting parsing of command script "offl2DFM7"                3
   :
  Starting Catalog Database verification
    Catalog Database verified
  Ending Catalog Database verification


  Starting Instance Manager verification
    Found Instance Manager (brd) process running with pid = 21228
  Ending Instance Manager verification


  Starting Target Database state verification
    Connecting to Target Database                               4
          ORACLE_SID: DFM7
       ORACLE_HOME: /u01/app/oracle/product/734
            DBNAME: DFM7
          Username: ebudba
      Connect String: dfm7

    Connected to idle instance.
    Issuing Statement: STARTUP PFILE=?/dbs/init@.ora DBA MOUNT;
    Database mounted.
    Target database state OK
  Ending Target Database state verification


  Database Backup job 9 started on 07-FEB-98 18:13              5

    Starting Database configuration verification


      No registration update is needed
      Target Database information is up to date in catalog

    Ending Database configuration verification


    Starting Database job building, involved files are:         6

      Parameter file "/u01/app/oracle/product/734/dbs/initDFM7.ora"

      Control file "/u01/oradata/DFM7/control01.ctl"

      Datafile "/u01/oradata/DFM7/system01.dbf"
      Datafile "/u01/oradata/DFM7/rbs01.dbf"
      Datafile "/u01/oradata/DFM7/temp01.dbf"
      Datafile "/u01/oradata/DFM7/tools01.dbf"
      Datafile "/u01/oradata/DFM7/users01.dbf"
      Datafile "/u01/oradata/DFM7/test01.dbf"
      Datafile "/u02/oradata/DFM7/test02.dbf"
      Datafile "/u03/oradata/DFM7/test03.dbf"

    Ending Database job building

    Starting Shutdown of target database for offline backup     7
      Issuing Statement: SHUTDOWN IMMEDIATE;
      Database dismounted.
      Database shutdown.
    Ended Shutdown of target database for offline backup
```

Figure 52 (Part 2 of 4). Sample EBU Offline Database Backup Log

```
  Starting backup to tape                                          8

    Number of parallel I/O streams: 2
                Oracle block size: 4096 bytes
            Disk input/output size: 65536 bytes
             Maximum tape I/O size: 131072 bytes
          Buffer size per I/O stream: 524288 bytes

    Starting BFS "22891998040" on 07-FEB-98 18:14 (40.004 MB)
      "/u01/oradata/DFM7/system01.dbf" for 40.004 MB
    Starting BFS "23891998041" on 07-FEB-98 18:14 (25.004 MB)
      "/u01/oradata/DFM7/rbs01.dbf" for 25.004 MB
⋮
    Starting BFS "31891998042" on 07-FEB-98 18:15 (0.049 MB)
      "/u01/oradata/DFM7/control01.ctl" for 0.049 MB
    Finished BFS "29891998041" (3 secs, 0.668 MB/s) on 07-FEB-98 18:15

    Finished BFS "31891998042" (2 secs, 0.025 MB/s) on 07-FEB-98 18:15


  Ending backup to tape

  Issuing Statement: STARTUP PFILE=?/dbs/init@.ora DBA MOUNT;
  Database mounted.
  This backup spawns archive sequence numbers 87 to 87
  This backup comprises SCN 9743 to 9743
  Issuing Statement: SHUTDOWN IMMEDIATE;
  Database dismounted.
  Database shutdown.

Database Backup job 9 SUCCESSFUL on 07-FEB-98 18:15                9


WARNING: Archivelog Files have not been backed up                  10
Use BACKUP ONLINE ARCHIVELOG command to backup archivelog files in another job
Recovery to current point might not be possible if archivelog files are lost



Catalog Backup job 10 started on 07-FEB-98 18:15                   11

  Catalog Database: "CAT7"

  Starting Catalog Backup job building
    Sequence 2 assigned to Backup of Catalog Database "CAT7"
      for today (07-FEB-98)
    Catalog will be backed up to file:
      "/u01/app/oracle/product/734/obackup/admin/CAT7247_02"
  Ending Catalog Backup job building


  Starting backup to tape

    Number of parallel I/O streams: 2
            Disk input/output size: 4096 bytes
             Maximum tape I/O size: 1024 bytes
          Buffer size per I/O stream: 4096 bytes

    Starting BFS "CAT7247_02" on 07-FEB-98 18:15 (0.008 MB)
      "/u01/app/oracle/product/734/obackup/admin/CAT7247_02" for 0.008 MB
    Finished BFS "CAT7247_02" (1 secs, 0.008 MB/s) on 07-FEB-98 18:15


  Ending backup to tape
```

Figure 52 (Part 3 of 4). Sample EBU Offline Database Backup Log

```
    Successful BACKUP of the Catalog Database "CAT7" with sequence 2
    Temporary Catalog backup file "/u01/app/oracle/product/734/obackup/admin/CAT7247_02" deleted

  Catalog Backup job 10 SUCCESSFUL on 07-FEB-98 18:15          12

BACKUP job SUCCESSFUL on 07-FEB-98 18:15                       13
```

Figure 52 (Part 4 of 4). Sample EBU Offline Database Backup Log

The following stages of the backup can be observed:

3 The backup script is parsed.

4 As the target database is shut down, it is mounted to perform the database verification.

5 The backup job starts.

6 The backup job compiles the list of files to be backed up.

7 The database is shut down to perform the offline backup.

8 Backup data is sent to tape by using two parallel I/O streams. The tapes are actually two sessions to the ADSM server.

9 A job result of SUCCESSFUL is posted for the backup.

10 A warning message is issued that the archive logs are not being backed up.

11 A separate catalog backup job is started. By default the catalog is backed up as an additional job after every EBU database backup job.

12 The catalog backup completes successfully.

13 The result of the overall backup process is posted.

7.6.3 Online Tablespace Backup

This example shows you how to perform online backup for a tablespace. This backup uses the default backup archive logs option and two parallel I/O streams. Figure 53 is a sample script called onlarcDFM7 that performs this online tablespace backup.

```
# script name:  onlarcDFM7
#
# Tasks:  EBU script to perform online backup for system tablespace
#         using 2 parallel I/O streams.
#
#

backup online tablespace=system          1
 db_name=DFM7
 parallel=2
 log=./onlarcDFM7.log
```

Figure 53. Sample Online Backup Script

The backup script contains a backup command for the system tablespace with the online parameter **1**.

The script is executed by using the **ebu** command with the script name as an option:

```
% ebu onlarcDFM7
```

Figure 54 shows extracts from the log generated by this backup.

```
Oracle7 Enterprise Backup Utility: Release 2.2.0.5.0 - Production on Tue Feb  3 13:24:10 1998

Copyright (c) Oracle Corporation 1979, 1996.  All rights reserved.

CORE Version 3.5.4.0.0 - Production
NLSRTL Version 3.2.4.0.0 - Production
   ⋮
BACKUP job started with pid=13724 on 02/03/1998 13:24:11
  Starting parsing of command script "onlarcDFM7"
   ⋮
    Datafile "/u01/oradata/DFM7/system01.dbf"              2

    Starting backup to tape

      Number of parallel I/O streams: 2
                  Oracle block size: 4096 bytes
              Disk input/output size: 65536 bytes
               Maximum tape I/O size: 131072 bytes
           Buffer size per I/O stream: 524288 bytes
      Issuing Statement: ALTER TABLESPACE "SYSTEM" BEGIN BACKUP;
      Statement processed.
   ⋮

    Ending backup to tape

    This backup spawns archive sequence numbers 1 to 1
    This backup comprises SCN 7209 to 7222

  Database Backup job 155 SUCCESSFUL on 03-FEB-98 13:27

  Archivelog Backup job 156 started on 03-FEB-98 13:27         3

    Starting ARCHIVELOG file job building:
      Archive log directory "/u01/app/oracle/product/734/dbs"
      Archive log file "/u01/app/oracle/product/734/dbs/arch1_95.dbf"
      Archive log file "/u01/app/oracle/product/734/dbs/arch1_98.dbf"
      Archive log file "/u01/app/oracle/product/734/dbs/arch1_96.dbf"
      Archive log file "/u01/app/oracle/product/734/dbs/arch1_97.dbf"
      Archive log file "/u01/app/oracle/product/734/dbs/arch1_1.dbf"
      WARNING: Some archived logs from LSN 1 to 98 are not backed up
    Ending ARCHIVELOG file job building                      4


    Starting backup to tape

      Number of parallel I/O streams: 1
                  Oracle block size: 4096 bytes
              Disk input/output size: 65536 bytes
               Maximum tape I/O size: 131072 bytes
           Buffer size per I/O stream: 524288 bytes

      Starting BFS "325886541235" on 03-FEB-98 13:27 (0.210 MB)
        "/u01/app/oracle/product/734/dbs/arch1_1.dbf" for 0.087 MB
        "/u01/app/oracle/product/734/dbs/arch1_97.dbf" for 0.004 MB
        "/u01/app/oracle/product/734/dbs/arch1_96.dbf" for 0.025 MB
        "/u01/app/oracle/product/734/dbs/arch1_98.dbf" for 0.077 MB
        "/u01/app/oracle/product/734/dbs/arch1_95.dbf" for 0.017 MB
```

Figure 54 (Part 1 of 2). Sample EBU Online Tablespace Backup Log

```
      Finished BFS "325886541235" (5 secs, 0.042 MB/s) on 03-FEB-98 13:27


    Ending backup to tape


  Archivelog Backup job 156 SUCCESSFUL (with WARNINGS)
  on 03-FEB-98 13:27                                          5


  Catalog Backup job 157 started on 03-FEB-98 13:27
    Catalog Database: "CAT7"
⋮

  Catalog Backup job 157 SUCCESSFUL on 03-FEB-98 13:27

BACKUP job SUCCESSFUL (with WARNINGS) on 03-FEB-98 13:27
```

Figure 54 (Part 2 of 2). Sample EBU Online Tablespace Backup Log

3 The system tablespace data file is identified and backed up.

3 After completion of the tablespace backup, an archive log job is started.

4 The archive log backup job warns about some missing archive logs.

5 The archive log backup completes with a warning about the missing archive logs. The overall backup of the tablespace also completes with a warning.

7.6.4 Multiplexed Online Database Backup

This example shows how to perform an online database backup with the multiplexing option. The multiplexing option is used to improve backup performance. Parallel sessions are also used whenever the multiplexing option is used. In this example the archive logs are deleted after they are backed up.

Figure 55 is a sample script called muxDFM7 that performs an online, multiplexed database backup.

```
# script name:   muxDFM7
#
#tasks:
#
# Using the multiplexing option, over two parallel I/O streams to ADSM
# archive log backed up and deleted afterwards.
#
#
backup online database
 db_name=DFM7
 archdelete                                                  1
 parallel=2                                                  2
 log=./onlmux.log
mux=(    "/u01/oradata/DFM7/rbs01.dbf",                     3
         "/u01/oradata/DFM7/system01.dbf",
         "/u01/oradata/DFM7/temp01.dbf",
         "/u01/oradata/DFM7/tools01.dbf",
         "/u01/oradata/DFM7/users01.dbf"
        )

mux=(     "/u01/oradata/DFM7/test01.dbf",                   3
          "/u02/oradata/DFM7/test02.dbf",
          "/u03/oradata/DFM7/test03.dbf"
        )
```

Figure 55. Sample EBU Script for Multiplexed Online Database Backup

The backup command performs an online database backup with the following additional options:

1 The archive logs are deleted after being backed up.

2 Two parallel data streams are used.

3 Each BFS will contain the data files within the ( ) for each of the two mux statements. The number of concurrent BFS streams depends on the number of parallel sessions defined.

The script is executed by using the **ebu** command with the script name as an option:

```
% ebu muxDFM7
```

Figure 56 illustrates extracts from the log file for the backup job.

```
Oracle7 Enterprise Backup Utility: Release 2.2.0.5.0 - Production on Tue Feb  3 15:20:25 1998

Copyright (c) Oracle Corporation 1979, 1996.  All rights reserved.

CORE Version 3.5.4.0.0 - Production
NLSRTL Version 3.2.4.0.0 - Production
   ⋮

BACKUP job started with pid=6394 on 02/03/1998 15:20:25

  Starting parsing of command script "muxDFM7"
  ⋮
    Starting backup to tape

      Number of parallel I/O streams: 2
                  Oracle block size: 4096 bytes
             Disk input/output size: 65536 bytes
              Maximum tape I/O size: 131072 bytes
          Buffer size per I/O stream: 524288 bytes
      Issuing Statement: ALTER TABLESPACE "USERS" BEGIN BACKUP;
      Statement processed.
      Issuing Statement: ALTER TABLESPACE "TOOLS" BEGIN BACKUP;
      Statement processed.
      Issuing Statement: ALTER TABLESPACE "TEMP" BEGIN BACKUP;
      Statement processed.
      Issuing Statement: ALTER TABLESPACE "SYSTEM" BEGIN BACKUP;
      Statement processed.
      Issuing Statement: ALTER TABLESPACE "RBS" BEGIN BACKUP;
      Statement processed.
      Starting BFS "327886548036" on 03-FEB-98 15:20(91.559 MB)  4
        "/u01/oradata/DFM7/users01.dbf" for 1.004 MB
        "/u01/oradata/DFM7/tools01.dbf" for 25.004 MB
        "/u01/oradata/DFM7/temp01.dbf" for 0.543 MB
        "/u01/oradata/DFM7/system01.dbf" for 40.004 MB
        "/u01/oradata/DFM7/rbs01.dbf" for 25.004 MB
      Issuing Statement: ALTER TABLESPACE "TESTMUX" BEGIN BACKUP;
      Statement processed.
      Starting BFS "328886548037" on 03-FEB-98 15:20 (3.012 MB)  5
        "/u01/oradata/DFM7/test03.dbf" for 1.004 MB
        "/u02/oradata/DFM7/test02.dbf" for 1.004 MB
        "/u03/oradata/DFM7/test01.dbf" for 1.004 MB
      Finished BFS "327886548036" (221 secs, 0.414 MB/s) on 03-FEB-98 15:24
```

Figure 56 (Part 1 of 2). Sample EBU Multiplexed Database Backup Log

```
     Issuing Statement: ALTER TABLESPACE "USERS" END BACKUP;
     Statement processed.
 ⋮
   Ending backup to tape
Database Backup job 169 SUCCESSFUL on 03-FEB-98 15:24


Archivelog Backup job 170 started on 03-FEB-98 15:24
 ⋮
   Ending backup to tape                                          6

   Archive log file "/u01/app/oracle/product/734/dbs/arch1_2.dbf" deleted
   Archive log file "/u01/app/oracle/product/734/dbs/arch1_97.dbf" deleted
   Archive log file "/u01/app/oracle/product/734/dbs/arch1_96.dbf" deleted
   Archive log file "/u01/app/oracle/product/734/dbs/arch1_98.dbf" deleted
   Archive log file "/u01/app/oracle/product/734/dbs/arch1_95.dbf" deleted

Archivelog Backup job 170 SUCCESSFUL (with WARNINGS) on 03-FEB-98 15:25


Catalog Backup job 171 started on 03-FEB-98 15:25
   Catalog Database: "CAT7"
 ⋮
```

Figure 56 (Part 2 of 2). Sample EBU Multiplexed Database Backup Log

4 The first BFS is created and sent to ADSM. It contains the data files defined in the first mux statement in the EBU script.

5 The second BFS is created and sent to ADSM. It contains the data files defined in the second mux statement in the EBU script.

6 On completion of the archive logs backup, the logs are deleted.

7.6.5 Recovering a Data File

This example shows how to restore a single data file. By default the archive logs are restored as well to their original location. After the restore operation is complete, the restored data file must be recovered.

Figure 57 is a sample script called restdfDFM7 that restores a single data file. In this example no automatic recovery is performed following the restore.

```
# script name:   restdfDFM7
#
# Tasks:
#
# EBU script to perform restore for single database datafile
# The archive logs will be restored to their original location
#
#

restore
 dbfile="/u01/oradata/DFM7/test01.dbf"                          1
 db_name=DFM7
 log=./restdfDFM7.log
```

Figure 57. Sample EBU Script for Restoring a Data File

The restore command specifies a single data file to be restored **1**.

Before the restore can be performed, the tablespace containing the data file must be offline. In this example the testmux tablespace is varied offline by using the Oracle server manager:

```
SVRMGR> alter tablespace testmux offline;
Statement processed.
```

After the tablespace is varied offline, the restore script is executed by using the **ebu** command with the script name as an option:

```
% ebu restdfDFM7
```

Figure 58 show extracts from the data file restore job.

```
Oracle7 Enterprise Backup Utility: Release 2.2.0.5.0 - Production on Tue Feb  3 15:43:18 1998

Copyright (c) Oracle Corporation 1979, 1996.  All rights reserved.

CORE Version 3.5.4.0.0 - Production
NLSRTL Version 3.2.4.0.0 - Production
  ⋮
RESTORE job started with pid=18996 on 02/03/1998 15:43:20
  Starting parsing of command script "restdfDFM7"
  ⋮
  Database Restore job 172 started on 03-FEB-98 15:43

    Starting Database job building, involved files are:

      Backup File Set "328886548037" from backup job 169          2
        Datafile "/u01/oradata/DFM7/test01.dbf"

      Starting ARCHIVELOG file job building:                      3
        Archive log files will be restored to their original backup locations
        Archive log file "/u01/app/oracle/product/734/dbs/arch1_95.dbf" found in
        Backup File Set "331886548298" from backup job 170
        Archive log file "/u01/app/oracle/product/734/dbs/arch1_98.dbf" found in
        Backup File Set "331886548298" from backup job 170
        Archive log file "/u01/app/oracle/product/734/dbs/arch1_96.dbf" found in
        Backup File Set "331886548298" from backup job 170
        Archive log file "/u01/app/oracle/product/734/dbs/arch1_97.dbf" found in
        Backup File Set "331886548298" from backup job 170
        Archive log file "/u01/app/oracle/product/734/dbs/arch1_2.dbf" found in
        Backup File Set "331886548298" from backup job 170
      Ending ARCHIVELOG file job building


    Ending Database job building

    Starting restore to disk

      Number of parallel I/O streams: 1
                  Oracle block size: 4096 bytes
             Disk input/output size: 65536 bytes
              Maximum tape I/O size: 131072 bytes
          Buffer size per I/O stream: 524288 bytes

      Starting BFS "328886548037" on 03-FEB-98 15:43 (1.004 MB)  4
        "/u01/oradata/DFM7/test01.dbf" for 1.004 MB
      Finished BFS "328886548037" (182 secs, 0.006 MB/s) on 03-FEB-98 15:46


      Starting BFS "331886548298" on 03-FEB-98 15:46 (0.167 MB)  5
        "/u01/app/oracle/product/734/dbs/arch1_95.dbf" for 0.017 MB
        "/u01/app/oracle/product/734/dbs/arch1_98.dbf" for 0.077 MB
```

Figure 58 (Part 1 of 2). Sample EBU Restore Data File Log

```
        "/u01/app/oracle/product/734/dbs/arch1_96.dbf" for 0.025 MB
        "/u01/app/oracle/product/734/dbs/arch1_97.dbf" for 0.004 MB
        "/u01/app/oracle/product/734/dbs/arch1_2.dbf" for 0.043 MB
      Finished BFS "331886548298" (3 secs, 0.056 MB/s) on 03-FEB-98 15:46
    Ending restore to disk

    This restore will require Archive Logs starting with sequence 1

  Database Restore job 172 SUCCESSFUL on 03-FEB-98 15:46

RESTORE job SUCCESSFUL on 03-FEB-98 15:46
```

Figure 58 (Part 2 of 2). Sample EBU Restore Data File Log

2 The BFS on the ADSM server containing the data file to be restored is identified along with the backup job number that created it.

3 A list of archived logs to be restored is built with details of the BFS on the ADSM server and the backup job number that originally backed them up.

4 The data file is restored from the ADSM server.

5 The archived logs are restored from the ADSM server.

After the data file restore is complete, recovery of the data file must be performed before the tablespace can be put backup online:

```
SVRMGR> recover datafile '/u01/oradata/DFM7/test01.dbf'
Media recovery complete.
SVRMGR> alter tablespace testmux online;
Statement processed.
SVRMGR>
```

This example of manual recovery is shown to illustrate the steps involved. Recovery can be automated in EBU by specifying a **recover** command in the EBU script following the **restore** command. With the recover command specified, EBU performs automatic recovery of the data file following the restore and places the tablespace backup online.

7.6.6 Recovering a Database

This example shows how to restore a database to its most current state. The database is restored inconsistent and then automatically recovered through EBU.

Figure 59 is a sample script called restDFM7 that restores and recovers the database.

```
# script name:  restDFM7
#
# Tasks:
#
# EBU script to perform restore database to most current state.
#

restore database
 db_name = "DFM7"
 parallel = 2
 recover                                                    1
 log=./restptDFM7.log
```

Figure 59. Sample EBU Script to Restore and Recover Database

The recover option **1** is specified in the script to recover the database after the restore. The database should be shut down before this job is executed:

```
SVRMGR> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SVRMGR>
```

The recover is executed by using the **ebu** command with the script name as an option:

```
% ebu restDFM7
```

Figure 60 shows extracts from the log created by this job.

```
Oracle7 Enterprise Backup Utility: Release 2.2.0.5.0 - Production on Wed Apr  8 11:43:38 1998

Copyright (c) Oracle Corporation 1979, 1996.  All rights reserved.

CORE Version 3.5.4.0.0 - Production
NLSRTL Version 3.2.4.0.0 - Production
   ⋮
  Database Restore job 23 started on 08-APR-98 11:43

    Starting Database job building, involved files are:          2

      Backup File Set "37892055852" from backup job 15
        Datafile "/u01/oradata/DFM7/system01.dbf"
        Datafile "/u01/oradata/DFM7/rbs01.dbf"
        Datafile "/u01/oradata/DFM7/temp01.dbf"
        Datafile "/u01/oradata/DFM7/tools01.dbf"
        Datafile "/u01/oradata/DFM7/users01.dbf"
      Backup File Set "38892055853" from backup job 15
        Datafile "/u01/oradata/DFM7/test01.dbf"
        Datafile "/u02/oradata/DFM7/test02.dbf"
        Datafile "/u03/oradata/DFM7/test03.dbf"
      Backup File Set "39892055854" from backup job 15
        File "/u01/app/oracle/product/734/dbs/initDFM7.ora" will be restored as
        "/u01/app/oracle/product/734/dbs/initDFM7.ora.00023"
      Backup File Set "40892055854" from backup job 15
        Control file "/u01/oradata/DFM7/control01.ctl" will be restored as
        "/u01/app/oracle/product/734/dbs/control01.ctl.00023"
        Control file "/u01/oradata/DFM7/control02.ctl" will be restored as
        "/u01/app/oracle/product/734/dbs/control02.ctl.00023"
        Control file "/u01/oradata/DFM7/control03.ctl" will be restored as
        "/u01/app/oracle/product/734/dbs/control03.ctl.00023"

      Starting ARCHIVELOG file job building:
        Archive log files will be restored to their original backup locations
        No archive logs need to be restored to disk, all are accounted for
      Ending ARCHIVELOG file job building


    Ending Database job building


    Starting restore to disk                                     3

      Number of parallel I/O streams: 2
                  Oracle block size: 4096 bytes
             Disk input/output size: 65536 bytes
```

Figure 60 (Part 1 of 3). Sample EBU Restore and Recover Database Log

```
              Maximum tape I/O size: 131072 bytes
          Buffer size per I/O stream: 524288 bytes

      Starting BFS "37892055852" on 08-APR-98 11:44 (91.559 MB)
        "/u01/oradata/DFM7/system01.dbf" for 40.004 MB
        "/u01/oradata/DFM7/rbs01.dbf" for 25.004 MB
        "/u01/oradata/DFM7/temp01.dbf" for 0.543 MB
        "/u01/oradata/DFM7/tools01.dbf" for 25.004 MB
        "/u01/oradata/DFM7/users01.dbf" for 1.004 MB

      Starting BFS "38892055853" on 08-APR-98 11:44 (6.012 MB)
        "/u01/oradata/DFM7/test01.dbf" for 2.004 MB
        "/u02/oradata/DFM7/test02.dbf" for 2.004 MB
        "/u03/oradata/DFM7/test03.dbf" for 2.004 MB
      Finished BFS "38892055853" (4 secs, 1.503 MB/s) on 08-APR-98 11:44


      Starting BFS "39892055854" on 08-APR-98 11:44 (0.004 MB)
        "/u01/app/oracle/product/734/dbs/initDFM7.ora.00023" for 0.004 MB
      Finished BFS "39892055854" (1 secs, 0.004 MB/s) on 08-APR-98 11:44


      Starting BFS "40892055854" on 08-APR-98 11:44 (0.148 MB)
        "/u01/app/oracle/product/734/dbs/control01.ctl.00023" for 0.049 MB
        "/u01/app/oracle/product/734/dbs/control02.ctl.00023" for 0.049 MB
        "/u01/app/oracle/product/734/dbs/control03.ctl.00023" for 0.049 MB
      Finished BFS "40892055854" (1 secs, 0.148 MB/s) on 08-APR-98 11:44

      Finished BFS "37892055852" (46 secs, 1.990 MB/s) on 08-APR-98 11:44


    Ending restore to disk

    This restore will require Archive Logs starting with sequence 88

    Starting Automatic Recovery                                    4
      Issuing Statement: STARTUP PFILE=?/dbs/init@.ora DBA MOUNT;
      Database mounted.
      Issuing Statement: ALTER DATABASE RECOVER AUTOMATIC;
      Statement processed.
      Issuing Statement: ALTER DATABASE OPEN;
      Statement processed.
      Issuing Statement: SHUTDOWN NORMAL;
      Database closed.
      Database dismounted.
      Database shutdown.
      Automatic Recovery Successful.
    Ended Automatic Recovery.


  Database Restore job 23 SUCCESSFUL on 08-APR-98 11:46

RESTORE job SUCCESSFUL on 08-APR-98 11:46
```

Figure 60 (Part 2 of 3). Sample EBU Restore and Recover Database Log

The following steps can be observed in the log:

2 The restore job is built and the files required are listed.

3 The database files are restored by using two parallel data streams from the ADSM server.

4 After the restore the recovery process is started. After the recovery completes, the database is shut down.

This database recovery recovers the database to its most current state. The same process can be used to recover to a point in time by specifying a date and time with the restore command, using the to option, **to =** "01/27/1998 12:00", and a date and time with the recover command, **recover =** "01/27/1998 15:00".

7.7 Administration

The examples in this section demonstrate some uses of the **ebutool** command to report and delete EBU backup objects. Three examples are shown:

- Listing the target database configuration
- Listing backup jobs
- Deleting backup jobs

7.7.1 Target Database Configuration

The **ebutool** command can be used to list the target database configuration as registered in the backup catalog. This is done by using the -db_name and -configuration options:

```
% ebutool -db_name=DFM7 -configuration=current
```

Figure 61 shows the output of the command.

```
Oracle7 Enterprise Backup Utility Tool: Release 2.2.0.5.0 - Production on Wed Apr  8 12:56:05 1998

Copyright (c) Oracle Corporation 1979, 1996.  All rights reserved.



Oracle7 Enterprise Backup Utility Tool                          08-APR-98 12:56


               Current registered configuration for database DFM7
               ==================================================


        Database: DFM7
         Created: 03-FEB-98 15:04
       Blocksize: 4096
  Last Resetlogs: 324313535


                Instance: DFM7
              Oracle Home: /u01/app/oracle/product/734
                    Host: bering
                  Domain: WORLD
    Connection Information:
                    User: ebudba
     Password enabled and protected
           Connect string: dfm7


       Configuration as of: 07-APR-98 18:16


        Tablespace SYSTEM:
            Datafile /u01/oradata/DFM7/system01.dbf
              Permissions:    640      Size: 40M bytes
                    Group:   200      Owner: oracle (700)


        Tablespace RBS:
            Datafile /u01/oradata/DFM7/rbs01.dbf
              Permissions:    640      Size: 24M bytes
```

Figure 61 (Part 1 of 3). ebutool Target Database Configuration Report

```
                Group:   200      Owner: oracle (700)


        Tablespace TEMP:
            Datafile /u01/oradata/DFM7/temp01.dbf
              Permissions:    640      Size: 556K bytes
                    Group:   200      Owner: oracle (700)


        Tablespace TOOLS:
            Datafile /u01/oradata/DFM7/tools01.dbf
              Permissions:    640      Size: 24M bytes
                    Group:   200      Owner: oracle (700)


        Tablespace USERS:
            Datafile /u01/oradata/DFM7/users01.dbf
              Permissions:    640      Size: 1028K bytes
                    Group:   200      Owner: oracle (700)


        Tablespace TESTMUX:
            Datafile /u01/oradata/DFM7/test01.dbf
              Permissions:    640      Size: 2052K bytes
                    Group:   200      Owner: oracle (700)
            Datafile /u02/oradata/DFM7/test02.dbf
              Permissions:    640      Size: 2052K bytes
                    Group:   200      Owner: oracle (700)
            Datafile /u03/oradata/DFM7/test03.dbf
              Permissions:    640      Size: 2052K bytes
                    Group:   200      Owner: oracle (700)




        Controlfiles:

            Controlfile /u01/oradata/DFM7/control01.ctl
              Permissions:    640     Size: 51712 bytes
                    Group:   200      Owner: oracle (700)
            Controlfile /u01/oradata/DFM7/control02.ctl
              Permissions:    640     Size: 51712 bytes
                    Group:   200      Owner: oracle (700)
            Controlfile /u01/oradata/DFM7/control03.ctl
              Permissions:    640     Size: 51712 bytes
                    Group:   200      Owner: oracle (700)


        Parameter Files:
            Parameter File /u01/app/oracle/product/734/dbs/initDFM7.ora
              Permissions:    644     Size: 4663 bytes
                    Group:   200      Owner: oracle (700)


     All files have at least one backup for this configuration

Oracle7 Enterprise Backup Utility Tool finished
```

Figure 61 (Part 2 of 3). ebutool Target Database Configuration Report

The output lists all data files within the database and their backup status. In this case there is a backup for all of the data files.

7.7.2 Listing Backup Jobs

The **ebutool** command with the -job= option is used to list the EBU job history from the backup catalog. The option can be specified as -job=all to list all EBU jobs and will produce a very large report of all EBU backup and restore jobs that have been run. Alternatively a job number can be specified to examine individual jobs. The following command produces a report for job 15, which was a full online database backup:

```
% ebutool -job=15
```

Figure 62 on page 147 shows the output of this command.

```
Oracle7 Enterprise Backup Utility Tool: Release 2.2.0.5.0 - Production on Wed Apr  8 13:18:19 1998

Copyright (c) Oracle Corporation 1979, 1996.  All rights reserved.


Oracle7 Enterprise Backup Utility Tool                          08-APR-98 13:18

                            Information for job 15
                            ======================


        Job 15:
           BACKUP ONLINE DATABASE performed by oracle
               Started at: 08-APR-98 10:17
                 Ended at: 08-APR-98 10:18
                  Runtime: 2 minutes
              Job Outcome: SUCCESSFUL


      Parallel Tape Streams:        2
                    thread:         1
     Archive Log File range: 88 to 88
    Archive Log Destination: /u01/app/oracle/product/734/dbs/arch%t_%s.dbf


             Database: DFM7
             Instance: DFM7
          Oracle Home: /u01/app/oracle/product/734
                 Host: bering
     As configured on: 07-APR-98 18:16


        Backup File Set: 37892055852
                Tape Id: BACKUPPOOL
               Datafile: /u01/oradata/DFM7/system01.dbf
          of tablespace: SYSTEM
                   Size: 40M bytes
               Datafile: /u01/oradata/DFM7/rbs01.dbf
          of tablespace: RBS
                   Size: 25M bytes
               Datafile: /u01/oradata/DFM7/temp01.dbf
          of tablespace: TEMP
                   Size: 556K bytes
               Datafile: /u01/oradata/DFM7/tools01.dbf
          of tablespace: TOOLS
                   Size: 25M bytes
               Datafile: /u01/oradata/DFM7/users01.dbf
          of tablespace: USERS
                   Size: 1028K bytes


        Backup File Set: 38892055853
                Tape Id: BACKUPPOOL
               Datafile: /u01/oradata/DFM7/test01.dbf
          of tablespace: TESTMUX
                   Size: 2052K bytes
               Datafile: /u02/oradata/DFM7/test02.dbf
          of tablespace: TESTMUX
                   Size: 2052K bytes
               Datafile: /u03/oradata/DFM7/test03.dbf
          of tablespace: TESTMUX
                   Size: 2052K bytes


        Backup File Set: 39892055854
                Tape Id: BACKUPPOOL
         Parameter File: /u01/app/oracle/product/734/dbs/initDFM7.ora
                   Size: 4663 bytes
```

Figure 62 (Part 1 of 2). ebutool Job History Report

```
        Backup File Set: 40892055854
                Tape Id: BACKUPPOOL
            Controlfile: /u01/oradata/DFM7/control01.ctl
                   Size: 51712 bytes

Oracle7 Enterprise Backup Utility Tool finished
```

Figure 62 (Part 2 of 2). ebutool Job History Report

This report lists the details of the backup job. It shows when the job was run, details of the target database, and details of the BFSs created on the ADSM server. Each BFS is listed as a unique number along with the tape ID on which it is stored. With the ADSMConnect Agent, the tape ID is the name of the ADSM server storage pool to which it was initially written. The data files included in the BFS are also listed. This inventory on the ADSM server can be verified by issuing the following ADSM administrative select command to query the backup inventory on the ADSM server:

```
adsm> select NODE_NAME, LL_NAME, STATE, BACKUP_DATE from backups where
    FILESPACE_NAME like '%adsmorc' order by LL_NAME desc

NODE_NAME            LL_NAME                          STATE          BACKUP_DATE
------------------   ------------------   ------------------   ------------------
   ⋮
BERING               40892055854             ACTIVE_VERSION          1998-04-08
                                                                  10:34:22.000000
BERING               39892055854             ACTIVE_VERSION          1998-04-08
                                                                  10:33:22.000000
BERING               38892055853             ACTIVE_VERSION          1998-04-08
                                                                  10:33:17.000000
BERING               37892055852             ACTIVE_VERSION          1998-04-08
                                                                  10:33:15.000000
   ⋮
```

The individual backup objects are ordered by low level name (LL_NAME). The names correspond to the BFS names from the job report (Figure 62 on page 147).

ADSM Select Command

The administrative select command was introduced with ADSM Version 3. It is not available with Version 2 servers.

7.7.3 Deleting Backup Jobs

Individual EBU jobs can be deleted from the backup catalog and their associated backup object from the ADSM server by using the **ebutool** command with the -deljobid= option:

```
% ebutool -deljobid=15
```

Figure 63 on page 149 shows the output of this command.

```
Oracle7 Enterprise Backup Utility Tool: Release 2.2.0.5.0 - Production on Wed Apr  8 13:49:37 1998

Copyright (c) Oracle Corporation 1979, 1996.  All rights reserved.




                          *** WARNING ***

You have selected to remove job 15


All information about the deleted backup job(s) will be lost irreversibly
Backup File Set(s) from the media will also be marked as invalid irreversibly

A restore to points in time covered by this (these) job(s) will not be longer possible

Answer YES to the following prompt if you want the operation to continue
Any other answer will abort the delete operation

                             PROCEED? => yes



Oracle7 Enterprise Backup Utility Tool                         08-APR-98 13:49


                             Deleting EBU Job 15
                             ===================


        Deleting Job 15:
           BACKUP ONLINE DATABASE performed by oracle
              Started at: 08-APR-98 10:17
                Ended at: 08-APR-98 10:18
                 Runtime: 2 minutes
             Job Outcome: SUCCESSFUL


BFS "40892055854" of job 15 removed successfully from backup media
BFS "39892055854" of job 15 removed successfully from backup media
BFS "38892055853" of job 15 removed successfully from backup media
BFS "37892055852" of job 15 removed successfully from backup media


Oracle7 Enterprise Backup Utility Tool finished
```

Figure 63 (Part 1 of 2). ebutool Delete Job Output

This job first prompts for confirmation that the job should be deleted. When confirmed by answering yes, the job is deleted from the backup catalog and the ADSM server.

The deletion from the ADSM server can be verified by issuing the ADSM administrative select command to query the backup inventory on the ADSM server:

```
adsm> select NODE_NAME, LL_NAME, STATE, BACKUP_DATE from backups where
    FILESPACE_NAME like '%adsmorc' order by LL_NAME desc

NODE_NAME            LL_NAME                        STATE           BACKUP_DATE
------------------   ------------------     ------------------    ------------------
   ⋮
BERING               40892055854              INACTIVE_VERSION          1998-04-08
                                                                    10:34:22.000000
BERING               39892055854              INACTIVE_VERSION          1998-04-08
                                                                    10:33:22.000000
BERING               38892055853              INACTIVE_VERSION          1998-04-08
                                                                    10:33:17.000000
BERING               37892055852              INACTIVE_VERSION          1998-04-08
                                                                    10:33:15.000000
   ⋮
```

The deletion of the BFS by EBU results in the backup objects on the ADSM server being made inactive. These will be deleted the next time the expire inventory process is run on the ADSM server. This ability to expire the backup objects is based on the ADSM server policy being defined as discussed in 7.3.2.6, “Define Management Class” on page 117.

The above example is just one method of deleting old backups from the backup catalog and the ADSM server, and it is a manual process. An alternative method is to use **ebutool** with the -purgejobs option. This method can be used to purge all jobs older than a specified number of days. Running this type of job on a periodic basis can simplify the task of managing the EBU backup inventory.

7.7.4 Backup Catalog Recovery

The backup catalog is critical to the operation of EBU. It contains the history of all backup operations performed on the target databases. This historical information is required to restore and recover a target database. Loss of the backup catalog renders all target database backups worthless.

EBU by default backs up the backup catalog as an additional job after all target database backups. This section looks at recovery of the backup catalog from these backups. First we show an example of an EBU job that restores the backup catalog. Second we look at how the backup catalog should be restored when multiple target databases share the backup catalog.

7.7.4.1 Backup Catalog Restore

The backup catalog is restored by using a restore command within an EBU command script. It is backed up as the second part of a target database backup and must also be restored by the target database. The catalog backup and restore process differs from the target database backup and restore process in that the former is a logical operation, and the latter is a physical operation. For a target database, physical data files are backed up and restored. With the catalog database instance, the catalog tables are logically backed up and recovered.

An EBU command script on the target database is used to restore the most recent backup catalog backup performed by that target database to the backup catalog database instance. Figure 64 is a sample script called restcat that restores the backup catalog. It is executed on the target database (DFM7) and restores the backup catalog on CAT7.

```
# script name:  restcat
#
# Tasks: Restore the backup catalog from the DFM7 target database
#

restore
 catalog = CAT7                                          1
 log=./restcat.log
```

Figure 64. Sample EBU Backup Catalog Restore Script

The EBU command script contains a restore command with catalog and the catalog instance (CAT7) as options **1**.

The script is executed by using the **ebu** command with the script name as an option:

```
% ebu restcat
```

Figure 65 shows the logfile created by running the restore job.

```
Oracle7 Enterprise Backup Utility: Release 2.2.0.5.0 - Production on Tue Feb  3 15:48:35 1998

Copyright (c) Oracle Corporation 1979, 1996.  All rights reserved.

CORE Version 3.5.4.0.0 - Production
NLSRTL Version 3.2.4.0.0 - Production
  ⋮
RESTORE job started with pid=22372 on 02/03/1998 15:48:35
  Starting parsing of command script "restcat"
        # script name:  restcat
        #
        #Tasks: Restore the backup catalog from the DFM7 target database
        #

        restore
         catalog = CAT7
         log=./restcat.log

  Ending parsing of command script
  Starting Catalog Database verification
    Catalog Database verified
  Ending Catalog Database verification

  Catalog Restore job 173 started on 03-FEB-98 15:48

    Catalog Database: "CAT7"
    Found backup of Catalog "CAT7", sequence 2 of day 03-FEB-98,
    BFS "CAT7208_02"                                                    2
    WARNING: Catalog is not empty, current contents will be dropped     3
    Starting catalog drop                                               4
      Issuing Statement: @@@catdrp;
      Catalog successfully dropped.
    Ended catalog drop
    Starting catalog creation                                           5
      Issuing Statement: @@@catcre6;
      Catalog successfully created.
    Ended catalog creation
    Starting restore to disk

      Number of parallel I/O streams: 1
              Disk input/output size: 4096 bytes
               Maximum tape I/O size: 1024 bytes
          Buffer size per I/O stream: 4096 bytes

      Starting BFS "CAT7208_02" on 03-FEB-98 15:58
        "/u01/app/oracle/product/734/obackup/admin/CAT7208_02"          6
      Finished BFS "CAT7208_02" (180 secs) on 03-FEB-98 16:01

    Ending restore to disk
    Starting Catalog restoration                                        7
    Ending Catalog restoration


  Catalog Restore job 173 SUCCESSFUL on 03-FEB-98 16:02

RESTORE job SUCCESSFUL on 03-FEB-98 16:02
```

Figure 65. Sample Backup Catalog Restore Log

The stages involved in restoring the catalog differ from those of restoring a target database because of the logical nature of the operation.

2 The most recent backup of the backup catalog taken by DFM7 within the last 7 days is determined.

3 In this example we are restoring over an existing backup catalog. The restore job determines that the catalog has data within it and issues a warning message.

4 The existing catalog is dropped.

5 The catalog tables are re-created.

6 The catalog backup is restored to a temporary file location.

7 The backup catalog tables are logically restored by using the catalog backup restored to the temporary file location.

The backup catalog has now been restored and can be used.

7.7.4.2 Multiple Target Databases

The task of restoring the backup catalog is straightforward. However, in an environment where a single backup catalog is shared by multiple target databases, the process becomes more complex. Each catalog backup performed by a target database contains the total catalog database for all target databases using the catalog. Restoration of the catalog by any one of the target databases restores the catalog for all target databases.

Figure 66 illustrates two target databases sharing a backup catalog and backing up to an ADSM server. Each target database has a separate ADSM filespace with the default EBU filespace name of /adsmorc.

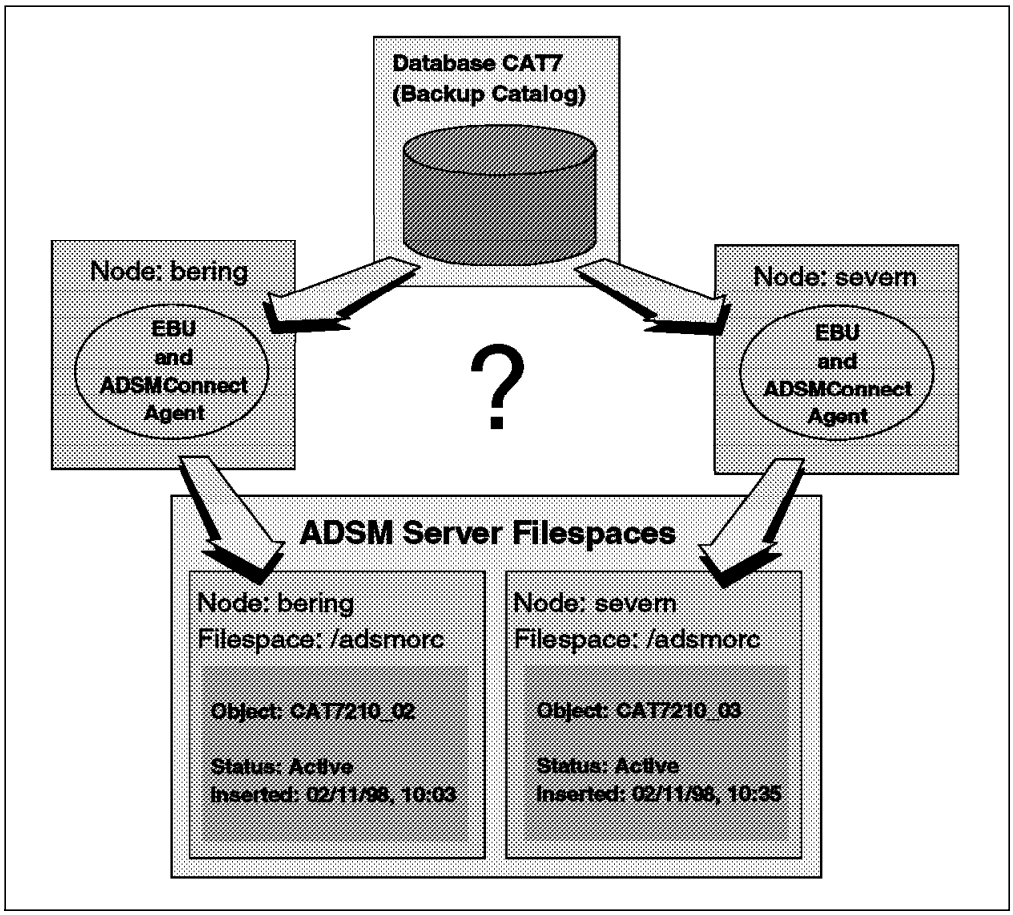


Figure 66. EBU Shared Backup Catalog Recovery

When EBU is used on the two target database nodes, bering and severn, the target database backups will also back up the shared backup catalog. However, EBU on each of the target database nodes, bering and severn, is unaware of each other. Successful recovery of the backup catalog in this environment is dependent on knowing which catalog backup to restore and on which target database to perform the restore. To restore the most current backup catalog, the restore must be performed on the target database that performed the most recent catalog backup. This backup will contain the most current information for all target databases that are using the catalog. Two possible approaches are:

1. Perform catalog backups from only one target database
2. Query the ADSM server to determine the latest catalog backup

A single target database could be nominated to perform all of the catalog backups. The other target databases could perform their backups by using the *catalog=none* option so that they do not back up the catalog. However, this scenario has some drawbacks. The target database performing the catalog backup is a single point of failure. The ability to recover the catalog is dependent on the availability of that target database instance. In addition, timing of the backup becomes critical. The target database performing the catalog backup must perform the backup after the other target databases to ensure that the backup will contain the latest information.

An alternative and simpler approach is to allow all of the target databases to back up the catalog and then determine from the ADSM server which is the most current. The catalog backups are held in the /adsmorc filespace for each node. The catalog backup object names within these filespaces always start with CAT. An ADSM administrative select command can be used to determine the latest catalog backup and on which node it was performed. Figure 67 illustrates an administrative macro that performs this task.

```
/* Query ADSM server to provide list of EBU catalog backups.*/

select NODE_NAME, LL_NAME as OBJECT_NAME, STATE, BACKUP_DATE from BACKUPS
where FILESPACE_NAME like '%adsmorc' and LL_NAME like 'CAT%' order by BACKUP_DATE desc
```

Figure 67. EBU Catalog Backup Query Macro

This macro lists all objects starting with CAT within a filespace containing adsmorc in its name. For each object found, the owning node name, the object's state, and the backup date will be displayed. The output will be ordered by descending date with the most current catalog backup at the top of the list.

Running the macro in Figure 67 from an ADSM administrative client produces the output shown in Figure 68 on page 154.

```
ADSTAR Distributed Storage Manager
Command Line Administrative Interface - Version 3, Release 1, Level 0.1
(C) Copyright IBM Corporation, 1990, 1997, All Rights Reserved.

Session established with server DB2: AIX-RS/6000
  Server Version 3, Release 1, Level 0.1
  Server date/time: 02/11/1998 15:24:58  Last access: 02/11/1998 15:24:49

ANS8000I Server command: 'select NODE_NAME, LL_NAME as OBJECT_NAME,
STATE, BACKUP_DATE from BACKUPS  where  FILESPACE_NAME like
'%adsmorc' and LL_NAME like 'CAT%' order by BACKUP_DATE desc'

NODE_NAME     OBJECT_NAME                 STATE        BACKUP_DATE
----------    -------------  ------------------ ------------------
SEVERN        CAT7210_03         ACTIVE_VERSION         1998-02-11  1
                                                      10:35:26.000000
BERING        CAT7210_02         ACTIVE_VERSION         1998-02-11
                                                      10:03:22.000000
BERING        CAT7210_00         ACTIVE_VERSION         1998-02-11
                                                      08:37:19.000000
   ⋮
BERING        CAT7200_03       INACTIVE_VERSION         1998-01-26
                                                      19:41:36.000000

ANS8002I Highest return code was 0.
```

Figure 68. EBU Catalog Backup Query Macro Output

From this output it can be seen that EBU on severn performed the most recent catalog backup **1**. Restoring the catalog on severn would recover the catalog to its most current state.

Chapter 8. Oracle8 Recovery Manager

This chapter describes how to use the Oracle8 Recovery Manager (RMAN) in conjunction with ADSM to back up and recover Oracle8 databases.

RMAN is Oracle's backup utility for Oracle8 databases. It provides functions to perform online and offline backups of the database, tablespaces, individual data files, control files, and redo logs. In addition it provides the capability to perform incremental database backups. These are backups of only those data blocks within data files that have changed data since the previous incremental backup. Restore and recovery can be performed for any component of the database. Roll forward recovery is performed by restoring the appropriate combination of incremental backups and redo logs and applying them to the restored database to the desired point in time.

RMAN provides the interface to the Oracle8 database and the functions for backup, restore, and recovery. It does not provide any storage management capabilities and must be integrated with other storage management products such as ADSM to provide a complete enterprisewide storage management solution. This integration is enabled through the ADSMConnect Agent for Oracle. The ADSMConnect Agent for Oracle is currently available for the following platforms:

- AIX 4.1 and above
- HP-UX 10.20 or 11.0
- Solaris 2.5.1

At the time of writing IBM is planning to introduce an ADSMConnect Agent for Oracle on Windows NT. The ADSMConnect Agent is the same agent as used with EBU and Oracle7.

This chapter is an introduction to RMAN and in particular how it integrates with ADSM. It is not an authoritative guide to RMAN or Oracle8 backup and recovery. This chapter should be read in conjunction with the *Oracle8 Server Backup and Recovery Guide*. The following topics are covered:

1. Introduction
2. Features
3. RMAN configuration
4. ADSMConnect Agent configuration
5. Using RMAN commands
6. Backup and recovery examples
7. RMAN administration
8. Recovery catalog availability

8.1 Introduction

RMAN is the backup and recovery tool provided by Oracle for Oracle8 Server databases and is packaged with the Oracle8 Server product. RMAN manages the process of backing up, restoring, and recovering Oracle8 target database instances. RMAN can automatically backup, restore, and recover the following database objects:

- Database
- Tablespaces

- Data files
- Control files
- Archived redo logs

In addition to backup and restore operations, RMAN can:

- Generate log records of all backup and recovery operations
- Run backup and restore operations in parallel to improve performance
- Find database objects that require a backup, on the basis of user-defined values

This section presents an overview of the architecture and the components involved in backing up Oracle8 databases with RMAN and ADSM.

8.1.1 RMAN and ADSMConnect Agent Architecture

To use RMAN to perform backups with ADSM, you have to integrate the ADSMConnect Agent into the Oracle8 Server product because RMAN relies on external media managers, such as ADSM, for storage management functions. Figure 69 shows how RMAN, running within an Oracle8 instance, connects through the ADSMConnect Agent to an ADSM server.

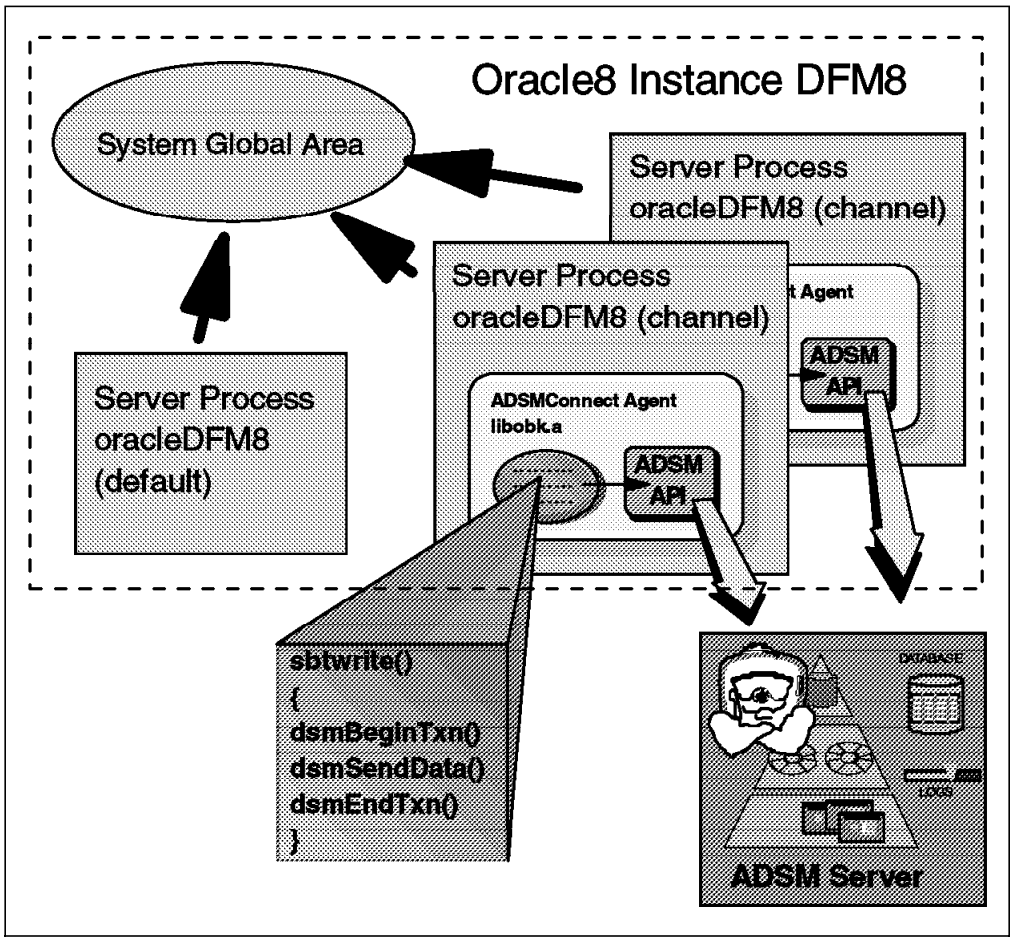


Figure 69. RMAN and ADSMConnect Agent Architecture

RMAN, when executed, starts a number of Oracle server processes on the target database. These Oracle processes read the database files to be backed up and send them by means of *channels* to an external media manager, in this case an

ADSM server. Channels are processes created through standard function calls defined in the Oracle media management, or SBT API. The ADSMConnect Agent provides these SBT function calls in a shared library called *libobk.a*. The ADSMConnect Agent also provides the standard ADSM API function calls to connect to an ADSM server.

The RMAN installation process links the Oracle8 server with this libobk.a library providing transparent connection from RMAN to an ADSM server. Once correctly installed and configured, RMAN processes can create channels to an ADSM server for backup and recovery purposes.

8.1.2 RMAN System Components

RMAN consists of several components that interact during the backup and recovery process.

8.1.2.1 RMAN Command

The RMAN command is the administrator's interface to RMAN. It invokes a command line interface that provides an operating-system-independent scripting language for performing backup and recovery operations. RMAN can be executed either interactively, where a command prompt is displayed and additional RMAN commands entered, or in batch mode, where an RMAN script containing commands is executed.

8.1.2.2 Target Database

The target database is the Oracle8 database instance on which RMAN executes specified backup, restore, and recovery actions. When the RMAN command is executed, it connects to the target database. The target database is specified by using RMAN parameters.

8.1.2.3 Communication Channel

RMAN can perform backup and restore functions to either local disk or to external media management products such as an ADSM server through the libobk.a library provided by the ADSMConnect Agent. These I/O operations are performed over a communication channel that defines the device to be used for the operation. The channel is used by RMAN to send or receive backup data to and from the I/O device.

For backup and restore operations, you must allocate a channel before the operation is performed. A channel corresponds to a single device. With the ADSMConnect Agent, a channel is a single session to an ADSM server. Multiple channels can be allocated. RMAN provides a multiplexing feature that enables parallel data streams to be sent over multiple allocated channels to maximize backup and recovery performance.

8.1.2.4 Recovery Catalog

The recovery catalog is the repository for information about backup objects created by RMAN. It is a Oracle8 database instance, separate from the target databases, and can contain information for multiple target databases. The data stored in the recovery catalog comprises structural information about the target databases to back up and restore. The recovery catalog contains information about:

- Physical schema of a target database

You have to register the target database at the recovery catalog to define the physical schema of the target database. RMAN needs to know about any structural change of the target database and obtains this information from the target database control file.

- Database backup history

  RMAN backs up databases, tablespaces, data files, control files, and archive logs to the ADSM server. Details of these backup objects held on ADSM is stored in the recovery catalog.

- Backup and recovery history

  RMAN stores backup, restore, and recovery information to maintain a history of previously performed operations. When backup and restore operations are performed, this information enables RMAN to determine:

  – Database files that require backing up
  – Old backup files that can be deleted
  – Files that are not recoverable

- Stored RMAN scripts

  RMAN commands can be stored in the recovery catalog as stored scripts. Scripts can be created to automate the execution of a several RMAN operations.

Oracle strongly recommends that a recovery catalog be used with RMAN. However, it is possible to execute RMAN commands without one. If you operate without a recovery catalog, RMAN uses the target database control file to store backup and structural information about the database. The following limitations apply when operating in this mode:

- Point-in-time recovery is not possible.
- Stored RMAN scripts cannot be used.
- Recovery cannot be performed if the control files are lost or damaged.

Oracle recommends the use of multiplexed control files, with each file located on different disks to protect against media failure.

Figure 70 on page 159 illustrates the system components involved in and the flow of an RMAN operation to ADSM.

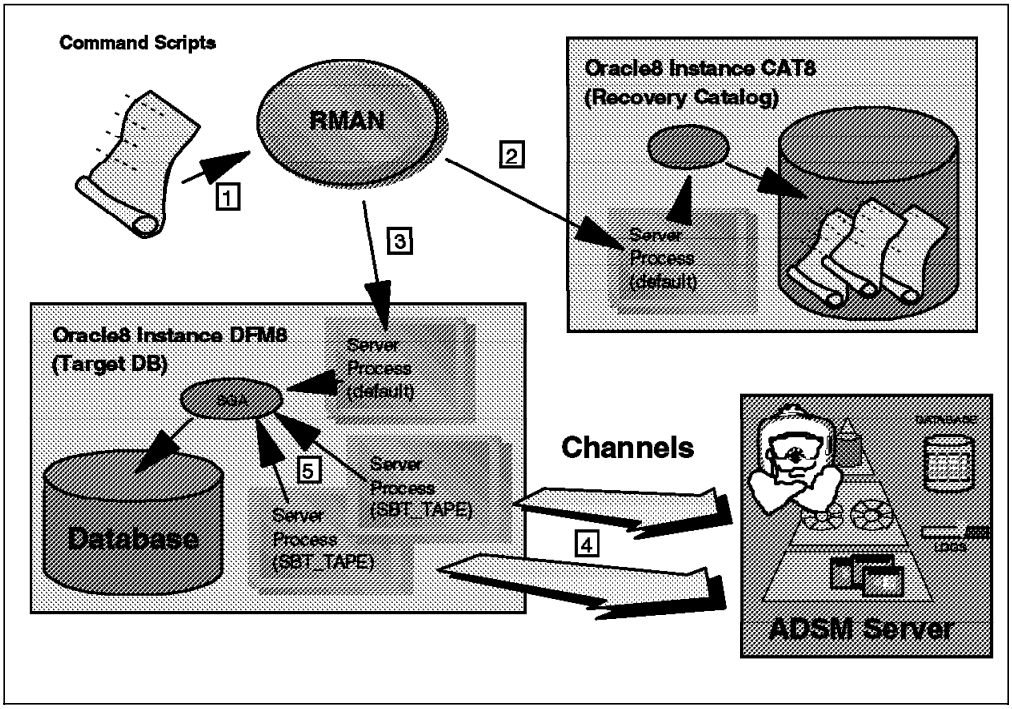


Figure 70. RMAN and ADSM System Components

The operation is invoked by invoking RMAN and entering the appropriate commands directly or by running a command script containing the commands **1**. RMAN connects to the recovery catalog **2** and the target database **3**. Before any backup or restore operations can be performed, RMAN allocates a channel to ADSM, using the SBT API **4**. RMAN then creates a server process on the target database instance **5** that performs the operation. For restore operations RMAN queries the recovery catalog to determine which files to restore from ADSM. For a backup operation RMAN backs up the objects specified in the command to ADSM. In both cases the data is transferred on the previously defined channel.

8.2 RMAN Features

This section introduces the features of RMAN operations. Topics covered are:

- Backup operations
- Restore operations
- Recovery operations
- Report generation
- Image copies

8.2.1 Backup Operations

Performing a backup with RMAN creates a backup set for that operation. A backup set contains backups of one or more data files or archive logs. Data file backup sets can also include control files. Archive log backup sets can contain only archive logs. Data file backup sets can be incremental or full and do not include empty blocks. A backup set is a logical entity composed of one or more physical output files called *backup pieces*. Each backup piece contains control and checksum information that allows the Oracle server process to validate the backup piece during a restore. A backup set is created by the **backup** command. A **restore** command is required to extract files from a backup set.

8.2.1.1 Full Backup

A full backup is a nonincremental backup of one or more data files. A full backup has no effect on incremental backups and is not considered to be part of the incremental strategy.

If the database is in ARCHIVELOG mode, you can choose to do full backup while the database is online or offline. If the database is in NOARCHIVELOG mode, the database must be closed by a clean shutdown. Full backups can be taken of:

- Data files
- Tablespaces
- Databases
- Control files
- Archive logs

8.2.1.2 Whole Database Backup

A whole database backup set contains the control files and all database files that belong to that database. Whole database backups do not require the database to be operated in a specific archiving mode. They can be taken whether a database is operating in ARCHIVELOG or NOARCHIVELOG mode. If the database is in ARCHIVELOG mode, you can choose to back up the database while it is open or closed. If running in NOARCHIVELOG mode, the database must be shut down first. There are two types of whole database backups:

- Consistent whole database backup

  A consistent whole database backup is a backup set where all files within it are consistent to the same point in time. A consistent whole database is the only valid backup for databases running in NOARCHIVELOG mode. The only way to take a consistent whole database backup is to shut down the database cleanly and take a backup while the database is offline.

- Inconsistent whole database backup

  An inconsistent whole database backup is a backup of an online database. It is inconsistent because portions of the databases may have been modified and written to disk during the backup process. The database must be in ARCHIVELOG mode in order to run an inconsistent backup.

  After an inconsistent backup is performed, the archived and online redo logs should also be backed up. Inconsistent whole database backups are restored and made consistent by applying any subsequent incremental backups and redo logs, online and archive, during the recovery process.

8.2.1.3 Incremental Backup

RMAN provides the capability of incrementally backing up databases at the individual block level. An incremental backup is a backup of one or more data files and contains only those blocks that have been modified since a previous backup at the same or lower level.

The *multilevel incremental backup* feature allows you to create different levels of incremental backups. Each level is denoted by an integer, with 0 being the lowest backup level. An incremental backup performed at a given level backs up only those blocks that have been modified since the last backup at the same or lower level. An incremental backup can be performed on:

- Individual data files
- Tablespaces
- The entire database

Incremental backup of control files or archived logs is not supported.

There are two types of incremental backups: *noncumulative* and *cumulative*.

Noncumulative Incremental Backup: A noncumulative incremental backup backs up only those blocks that have changed since the previous incremental backup at the same or lower level. This is the default mode of operation for incremental backups.

A level 0 backup backs up all blocks that contain data. It performs the same backup as a full backup. A level 0 backup is required for subsequent incremental backups at other levels. An incremental backup at a level greater than level 0 backs up only those blocks that have changed since a previous incremental backup at the same or lower level. The size of the backup depends on the number of blocks modified.

Figure 71 illustrates part of a monthly cycle of noncumulative incremental backups. The cycle is based on backup levels 0, 1, and 2. A monthly backup is performed at level 0 on the first Sunday of the month, daily incremental backups are performed at level 2, and weekly incremental backups are performed at level 1 on the remaining Sundays in the month.

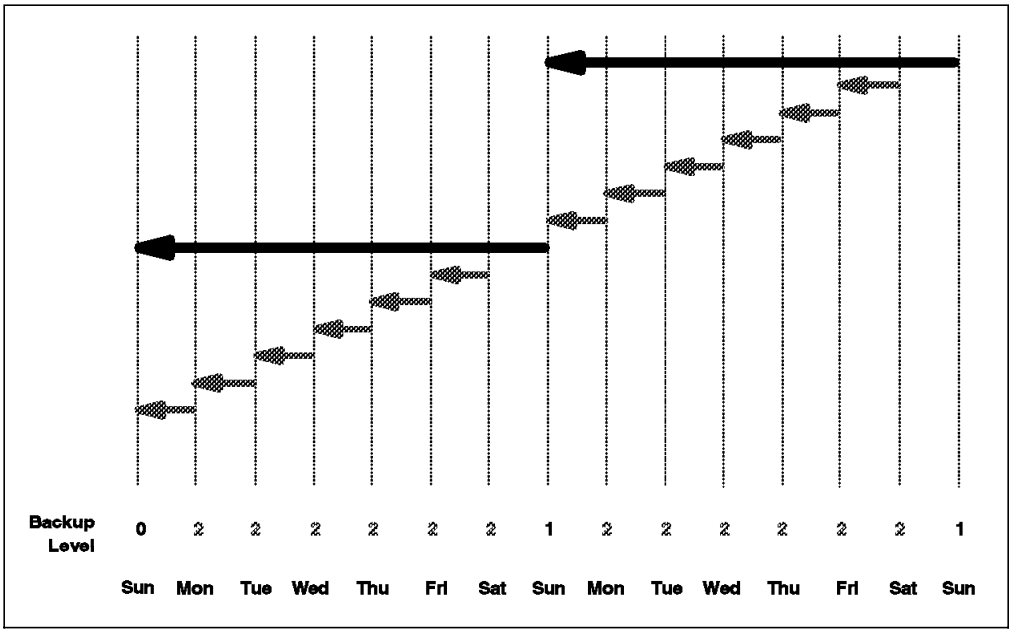


Figure 71. RMAN Noncumulative Incremental Backup

In Figure 71 all of the backups are noncumulative. They back up only those blocks that have changed since the previous backup at the same or lower level. Table 4 describes the backup operations that take place during the first week of this monthly cycle.

Table 4 (Page 1 of 2). RMAN Noncumulative Incremental Backup

Day	Backup Level	Backup Operation
First Sunday	0	All blocks that have ever been in use in this database are backed up. This is a full backup of the database.

Table 4 (Page 2 of 2). RMAN Noncumulative Incremental Backup

Day	Backup Level	Backup Operation
Monday to Saturday	2	All blocks that have changed since the most recent incremental backup at level 2 or lower are backed up. On the first Monday, any blocks changed since the Sunday level 0 backup are backed up. On subsequent days during the week only blocks that have changed since the previous level 2 backup are backed up. This is a daily incremental backup.
Remaining Sundays	1	All blocks that have changed since the most recent incremental backup at level 1 or lower are backed up. On the second Sunday in the month, the most recent backup at level 1 or lower is the level 0 Sunday backup, so all blocks changed since then will be backed up. This includes all blocks that have changed during the first week, including those backed up by the daily level 2 backups. This is a weekly incremental backup.

This cycle is repeated every week during the month. The weekly level 1 backs up the week's worth of changes, and the daily level 0 backs up only the day's worth of changes. With this method elapsed backup times can be reduced, particularly during the week. However, the recovery process is more complex. To recover to a particular point in time, the most current level 0 or 1 and subsequent level 2 backups must be restored.

Cumulative Incremental Backup: A cumulative incremental backup is a backup of one or more data files that contains only those blocks that have changed since a previous backup at a lower level. A cumulative backup is run by specifying the **cumulative** option when defining the backup job.

Figure 72 on page 163 illustrates a weekly cumulative incremental backup where full backups are performed on Sundays at level 0 and cumulative incremental backups are performed during the week at level 2.

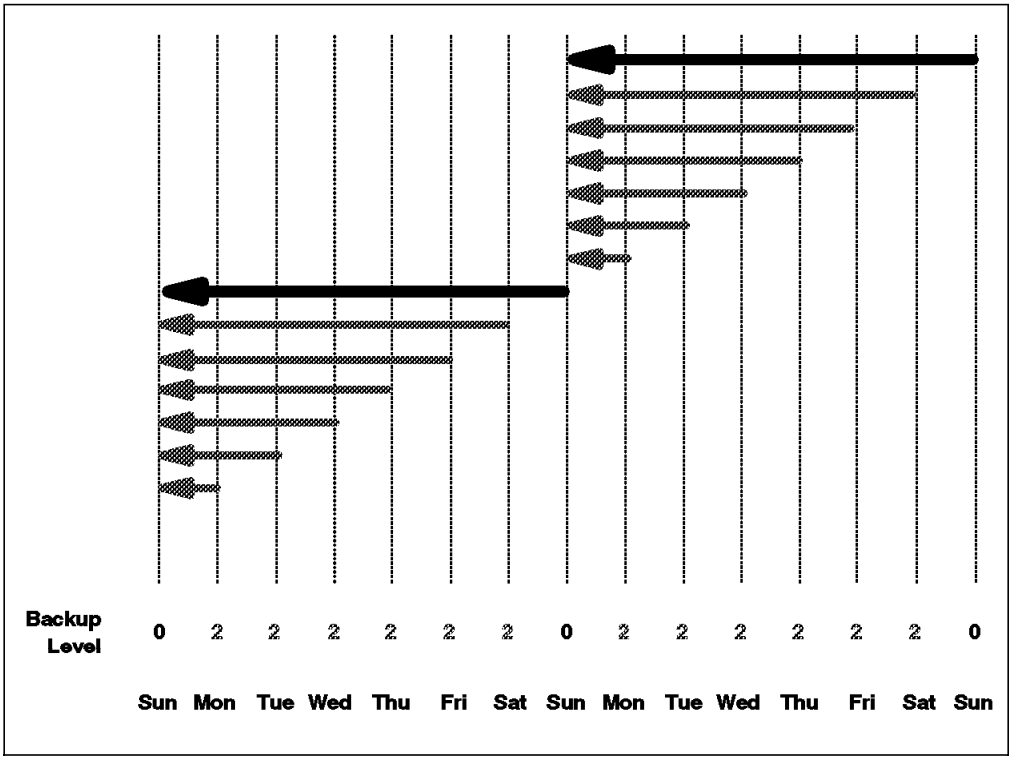


Figure 72. Cumulative Incremental Backup

In Figure 72 all of the level 2 backups are cumulative. They back up all blocks that have changed since the previous backup at a lower level. In this example the level 0 backup occurs on Sundays. Table 5 describes the backup operations that take place during this weekly cycle.

Table 5. RMAN Cumulative Incremental Backup

Day	Backup Level	Backup Operation
Sundays	0	All blocks that have ever been in use in this database are backed up. This is a full backup of the database.
Monday to Saturday	2	All blocks that have changed since the most recent incremental backup at a level lower than 2 are backed up. Every day all blocks that have changed since the Sunday level 0 backup are backed up. Each daily backup also includes those blocks backed up by previous level 2 backups.

A cumulative incremental backup reduces the time and the work needed for restore by ensuring that you need one incremental backup from any level at restore time. However, it requires more space for storing the data, and the backup process takes longer because it duplicates the work done by previous backups at the same level.

8.2.2 Restore Operations

Recovering a database, tablespace, or data file is a two-stage process. The object is first restored and then it is recovered.

The restore process restores the necessary full or incremental level 0 backups. Incremental backups at levels greater than 0 are not restored. These are restored during the subsequent recovery process.

By default the objects are restored to their original location as specified in the recovery catalog. An alternative location can also be specified if required. RMAN uses the recovery catalog to select the most current backup sets for use in the restore. The mode in which the database is running determines whether a *consistent* or *inconsistent* restore operation can be performed.

8.2.2.1 Consistent Database Restore

If the database in running in NOARCHIVELOG mode, it can be restored only from a whole database backup. The control files and all data files are restored from a consistent backup. The database must be mounted but not open during the restore operation.

After a consistent restore, the database can be opened without performing any recovery. Any updates to the database after the backup are lost.

8.2.2.2 Inconsistent Database Restore

If the database is running in ARCHIVELOG mode, a subset of the database such as a data file, tablespace, or the entire database can be restored to the most current state or to a specific point in time. This type of restore is inconsistent because the database cannot be started after the restore. The restore operation must be followed by a recovery operation. After the recovery is finished, the database can then be started. The following objects can be restored:

- Database
- Tablespace
- Data file
- Control file
- Archive log

If the control files are lost, they must be restored before other restore operations can be performed. Only after restoring the control files can the target database be mounted and the other restore operations started.

A restore operation is typically followed by a recovery operation.

8.2.3 Recovery Operations

Recovery is a process whereby a restored file is made available, either to the most current state or to a specific point in time.

Once the data files are restored, they have to be made consistent with each other. The **recover** command is used to perform media recovery and to apply incremental backups. During the recovery process, RMAN automatically restores the archived redo logs as required.

If RMAN has a choice between restoring incremental backup sets or applying redo logs, it always uses the incremental backups. If overlapping levels of

incremental backup are available, the lowest level of incremental backup, the one covering the longest period of time, is chosen automatically.

The following objects can be recovered:

- Individual data files

  One or more data files can be recovered. The target database must be started and mounted. If the target database is open, the data file must be offline.

- Tablespaces

  All data files in a tablespace can be recovered or a tablespace can be recovered to a previous point in time. The target database must be started and mounted. If the target database is open, the tablespace must be offline.

- The entire database

  The entire database can be recovered under the following circumstances:

  – A media failure has damaged the entire database.
  – The entire database must be recovered to a previous point in time.
  – The control files have been lost.

  For database recovery the database must be started but not open. If possible it should be mounted.

Archive log backup sets are restored as needed to perform a recovery. They are restored to the current archive log destination as specified in the *init.ora* file.

8.2.4 Report Generation

RMAN can generate reports about backups and images copies with the **report** and **list** commands.

The **report** command can produce the following types of information from the recovery catalog:

- The database schema
- Which data files require a backup
- Backup files that are obsolete and can be deleted
- Data files that are not recoverable

The **list** command queries the recovery catalog and produces information such as:

- List of backup sets containing a backup of a specific data file
- List of backup sets containing backups for a specific tablespace
- List of backup sets containing backups for the whole database

8.2.5 Image Copies

An image copy is a single file (data file, archive log, or control file) that can be used as-is to perform a recovery. It is similar to an operating system copy of a single file except it is produced by an Oracle server process that performs additional tasks such as validating the blocks in the file and registering the copy in the control files. An image copy can be done only to disk.

Image copies are not discussed further in this chapter.

8.3 RMAN Configuration

RMAN is distributed with the Oracle8 Server product and does not have to be installed separately. However, it must be configured.

To configure RMAN the following Oracle software products are required:

- Oracle Server (RDBMS) Version 8.0.3 or higher
- Oracle Server Manager or Enterprise Manager
- Oracle Net8 Version 8.0.3
- PL/SQL Version 8.0.3
- SQL/Plus Version 8.0.3

In addition, for RMAN to send backups to ADSM, the ADSMConnect Agent and one or more ADSM servers are required.

The following steps must be performed to configure RMAN for use with a recovery catalog:

1. Install the recovery catalog.
2. Start the listener process.
3. Create the recovery catalog tablespace.
4. Create the recovery catalog user.
5. Create the recovery catalog schema.
6. Register the target database.

The examples are based on two Oracle8 instances:

- Target database DFM8 (Oracle 8.0.3)
- Recovery catalog database CAT8 (Oracle 8.0.3)

8.3.1 Install Recovery Catalog

For the recovery catalog, an Oracle8 database instance separate from the target database is required. However, you do not need to have a database instance solely for the recovery catalog. The recovery catalog tables could be stored in another nontarget database Oracle database instance, although this is not recommended.

To install an Oracle8 database you use the Oracle8 installation utility **orainst**. The utility is covered in the *Oracle8 Installation Guide* and is not discussed further here.

8.3.2 Start the Listener Process

The listener process enables communication between all of the databases in an Oracle networked environment. It must be configured on all Oracle8 instances that will communicate. Follow these steps:

1. When Oracle8 is installed it creates an Oracle user account. This account is used for administration purposes. Switch to the Oracle user and ensure that the following environment variables are set:
   - ORACLE_HOME
   - ORACLE_SID
   - ORACLE_BASE
   - ORACLE_DOC
2. As the Oracle user, change to the $ORACLE_HOME/network/admin directory on the target database and ensure that the listener process is correctly

configured by checking the listener configuration file, listener.ora. The HOST and PORT should be defined in the LISTENER stanza, and the database instances being listened for should be defined in the SID_LIST_LISTENER stanza:

```
% cd $ORACLE_HOME/network/admin
% cat listener.ora

LISTENER =                ◄
 (ADDRESS_LIST =
  (ADDRESS =
(PROTOCOL = TCP)
(HOST = severn)           ◄
(PORT = 1521)             ◄
  )
 )

SID_LIST_LISTENER =       ◄
 (SID_LIST =
  (SID_DESC =
    (SID_NAME = DFM8)     ◄
    (ORACLE_HOME = /oracle/app/oracle/product/803 )
 (GLOBAL_NAME = DFM8)
   )
)
```

Check the same definitions on the Oracle8 instance being used for the recovery catalog.

3. Ensure that $ORACLE_HOME/network/admin/tnsnames.ora on both the target and recovery catalog databases has a connect string for each of the database instances that will be used:

```
% cat $ORACLE_HOME/network/admin/tnsnames.ora
  ⋮
       DFM8.almaden.ibm.com =           ◄
       (DESCRIPTION =
         (ADDRESS = (PROTOCOL = TCP)(HOST = severn)(PORT = 1521))
         (CONNECT_DATA =(SID = DFM8))
       )


       CAT8.almaden.ibm.com =           ◄
       (DESCRIPTION =
         (ADDRESS = (PROTOCOL = TCP)(HOST = black)(PORT = 1521))
         (CONNECT_DATA =(SID = CAT8))
       )
  ⋮
```

The name of the connect strings (DFM8, CAT8) are the user-definable aliases. The tnsnames.ora file must contain the connect strings for all of the databases you use. The PROTOCOL, PORT, HOST, and SID values depend on your environment. In our environment we used PROTOCOL TCP and PORT 1521.

4. Start the listener process.

Having checked that each database has its listener instance and connect string defined, start the listener process. You can use the following commands to manipulate the listener:

Start the listener process:

```
%lsnrctl start

   ⋮
Starting /u01/app/oracle/product/803/bin/tnslsnr: please wait...
   ⋮
  DFM8          has 1 service handler(s)
The command completed successfully
```

Check the listener process status:

```
%lsnrctl status

   ⋮
Connecting to (ADDRESS=(PROTOCOL=TCP)(HOST=severn)(PORT=1521))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
   ⋮
Listener Parameter File   //u01/app/oracle/product/803/network/admin/listener.ora
Listener Log File         /u01/app/oracle/product/803/network/log/listener.log
Services Summary...
  DFM8          has 1 service handler(s)
The command completed successfully
```

Stop the listener process:

```
% lsnrctl stop
%lsnrctl stop
   ⋮
(c) Copyright 1997 Oracle Corporation.  All rights reserved.

Connecting to (ADDRESS=(PROTOCOL=TCP)(HOST=severn)(PORT=1521))
The command completed successfully
```

Before you continue make sure that you can access the recovery catalog from each target database. The **tnsping** command can be used to confirm this:

```
% tnsping CAT8
   ⋮
Attempting to contact (ADDRESS=(PROTOCOL=TCP)(HOST=black)(PORT=1521))
OK (100 msec)
%
```

8.3.3 Create Recovery Catalog Tablespace

You must create a separate tablespace on the recovery catalog database instance for the recovery catalog information. The size of the tablespace depends on how much database backup records you want to keep. For our examples we created a 20 MB tablespace called **rcvcat**. As the Oracle user we connected to the recovery catalog database, CAT8, using **svrmgrl**, and created a tablespace:

```
% svrmgrl system/manager@CAT8
svrmgrl>connect internal ;
svrmgrl>create tablespace rcvcat datafile 'rcvcat.dbf' size 20M ;
```

8.3.4 Create Recovery Catalog User

An Oracle8 user must be created that will own the recovery catalog. As the Oracle user, connect to the Oracle8 recovery catalog instance and create the user. In our example we used a user named cat8dba with a password of cat8dba. The temporary tablespace for this user is temp and the default tablespace is rcvcat:

```
%svrmgrl
SVRMGR>connect internal;
SVRMGR>create user cat8dba identified by cat8dba
     2>temporary tablespace temp
     3>default tablespace rcvcat
     4>quota unlimited on rcvcat;
Statement processed.
```

The cat8dba user is granted authority as the recovery catalog owner:

```
SVRMGR>
SVRMGR>grant recovery_catalog_owner to cat8dba;
Statement processed.
```

8.3.5 Create Recovery Catalog Schema

Having created the recovery catalog tablespace and user, you must set up the recovery catalog schema. Oracle8 provides a script in $ORACLE_HOME/rdbms/admin/ called **catrman.sql** to perform this task.

Connect to the Oracle recovery manager database as that user and execute the **catrman.sql** script to create the catalog schema:

```
SVRMGR>connect cat8dba/cat8dba
SVRMGR>@?/rdbms/admin/catrman
Statement processed.
   :
   :
   :
Statement processed.
SVRMGR>
```

The script creates the recovery catalog schema in the cat8dba user's default tablespace, rcvcat, previously created. This script will display a number of warning messages as it first attempts to drop the objects it will create. This is done to ensure that a clean schema is created.

On completion the script will have created all of the recovery catalog objects, and it is now ready for use.

8.3.6 Register the Target Database

Having created a recovery catalog, you must register the target databases to it. Use RMAN commands. Registration must be done before any other RMAN functions can be performed on the target databases.

You register each of the target database instances that will use the recovery catalog. As the Oracle user run the **rman** command to connect to the target and recovery catalog databases and then run the **register database** command.

The following example shows the commands to connect to the DFM8 target database as the internal user and the CAT8 recovery catalog database as the newly created cat8dba user, followed by the target database being registered:

```
> su - oracle
% rman target internal/manager@DFM8 rcvcat cat8dba/cat8dba@CAT8

Recovery Manager : Release 8.0.3.0.0 - Production
RMAN-06005:  connected to target database DFM8
RMAN-06008:  connected to recovery catalog database

RMAN>register database;
  ⋮
RMAN>exit

Recovery Manager Complete.
%
```

The target database instance (DFM8) is now registered with the recovery catalog, and RMAN is now configured for that target database. At this stage RMAN commands can be executed to perform backup, restore, and recovery functions for the target database. However, these functions can be performed only to local disk files. The ADSMConnect Agent must be installed and configured to enable RMAN functions to use ADSM.

8.4 ADSMConnect Agent

This section covers implementation of the ADSMConnect Agent for use with RMAN.

8.4.1 Installation

This section covers installation of Versions 2.1.0.6 and 2.1.0.7 of the ADSMConnect Agent for AIX. To install the ADSMConnect Agent, use SMIT or the AIX **installp** command. The following file sets must be installed:

adsmagent.aob.obj
adsmagent.aobord.obj

The ADSMConnect Agent installation path is */usr/lpp/adsmagent/aob.* Table 6 lists the files installed. The files differ slightly for the different versions.

Table 6 (Page 1 of 2). ADSMConnect Agent Files

	Description
adsmaob.ord	Version 2.1.0.6 verification file
agent.lic	Version 2.1.0.7 verification file
dsm.opt.smp	Sample client user options file
dsm.sys.smp	sample client system options file
en_US/dsmclient.cat	Version 2.1.0.6 message catalog
en_US/dsmclientv3.cat	Version 2.1.0.7 message catalog

Table 6 (Page 2 of 2). ADSMConnect Agent Files	
	Description
libobk.a	Shared library
aobpswd	Password manager program
options.doc	Documentation for client options
README.AOB	Readme file

One of the main differences between the two levels of the ADSMConnect Agent is the way that the libobk.a library is built. The libobk.a library provided with Version 2.1.0.6 of the ADSMConnect Agent has Version 2.1.0.6 of the ADSM API statically linked within it. Therefore separate installation of the ADSM API client is not necessary. The libobk.a library with Version 2.1.0.7 of the ADSMConnect Agent is different. It is dynamically linked to Version 3.1.3 of the ADSM API client, and Version 3.1.3 or later of the API client must be installed separately.

As part of the ADSMConnect Agent installation, a symbolic link is created from */usr/lib/libobk.a* to *libobk.a* in the ADSMConnect Agent library.

8.4.2 Configuration

Having installed the ADSMConnect Agent, you must configure it. This involves updating the dsm.sys and dsm.opt files, registering the nodename on the server, configuring environment variables, and setting the ADSMConnect Agent password.

8.4.2.1 Edit Options Files

The dsm.sys and dsm.opt files must be created with the correct communications options to connect to the ADSM server.

Ensure that the PASSWORDACCESS option is not set to generate in the dsm.sys file (prompt is the default if not specified). If it is set to generate, RMAN will not work because the generate function forks a separate process that conflicts with Oracle′s message-passing activities. Using prompt does not fork a process.

8.4.2.2 Register Node

Register the node on the ADSM server with the BACKDELETE=yes option. The system′s hostname will be used by default as the nodename if one is not defined in the dsm.sys file.

8.4.2.3 Initialize Password

The ADSMConnect Agent provides a program, aobpswd, to simulate PASSWORDACCESS=GENERATE. The program eliminates the need to manually enter a password, allowing RMAN to contact the ADSM server without prompting each time for the password. The aobpswd program encrypts the ADSM password and stores it in a secure file located with the DSMO_PSWDPATH environment variable. The password is then passed by the ADSM API to the ADSM server without user intervention.

Note

A key difference between the aobpswd program and passwordaccess generate is that the aobpswd program does not automatically update the password when it expires, that is, it does not automatically regenerate the password. The backup jobs will fail when the passwords are no longer valid. Password changes should be scheduled before expiration has occurred.

Before using the ADSMConnect Agent for Oracle, you must run aobpswd as the root user.

To initialize the ADSMConnect Agent password follow these steps:

1. Start the aobpswd program to set up the connection with the server that you specified in the dsm.opt file.
2. Enter the current node password. You are prompted for a new password. You do not have to enter any value; just press Enter. The following message is displayed:

   ```
   Your password has been written to the File.
   Verify that the DBA has read access to the
   ADSMO.yourhostname file.
   ```

   Make sure that the user who will run the ADSMConnect Agent has read permission to this file.

You also use aobpswd to update the password when it expires.

8.4.2.4 Customize the Management Class

As both the backup-archive and API clients can be used on the same node, it is recommended that separate management classes be used for the different file systems.

RMAN manages it own inventory of backups, using the ADSMConnect Agent and the ADSM server as a secure repository to maintain them. RMAN determines the names of the individual objects that are stored on ADSM. Each object has a unique name, so each successive backup of the same data has a different name on the ADSM server. Therefore only one copy of an object from RMAN is ever stored on the ADSM server. All RMAN objects stored on the ADSM server remain active (because they are the only copy) and can be individually deleted only when RMAN issues a delete object API call. When RMAN has deleted the object it can be discarded on the ADSM server.

A separate management class should be created with a backup copy group (the ADSMConnect Agent does not use archiving) defined that maintains only one active version of the backup object and has the following parameters for deleting expired backup objects:

VERDELETE=0
RETONLY=0

With a unique name only one copy of the backup object will be held on the ADSM server. When RMAN deletes the object from the recovery catalog and ADSM, it issues a delete object API call to change the status of the object from active to inactive on the ADSM server. This inactive object will be expired the next time that inventory expiration is run.

When a database is backed up, the default management class for the node name will be used unless you override it with a different value in an include-exclude file by adding an INCLECXL option to the dsm.sys file that points to the client include/exclude list. The following sample statement in the include-exclude file binds all backups belonging to the ADSMConnect Agent filespace, */adsmorc*, to the ORACLE management class:

```
INCLUDE /adsmorc/.../* ORACLE
```

8.4.3 Linking ADSMConnect Agent to RMAN

It is important to install the ADSMConnect Agent on the system containing the target database before using RMAN with ADSM. The ADSMConnect Agent contains the shared library libobk.a and creates a symbolic link to /usr/lib/libobk.a. Because Oracle links the ADSM API shared library libobk.a to connect to ADSM, the shared library must exist before relinking the Oracle Server product.

After installing the ADSMConnect Agent on the system containing the target database, you must link the Oracle software with the shared library. Before you link the Oracle software, you must shut down all databases. The linking operation enables Oracle8 Server processes to connect to the ADSMConnect Agent. You can relink the Oracle8 Server product by using the Oracle8 installer utility orainst. For information about this method refer to Chapter 4, ″Configuring the Oracle8 System,″ in the *Oracle Installation Guide*. The Oracle8 Server product can be manually relinked by using the library manager parameter, LLIBMM=/usr/lib/libobk.a. As the Oracle user enter the following at an AIX command prompt:

```
% cd $ORACLE_HOME/rdbms/lib
% make -f ins_rdbms.mk ioracle LLIBMM=/usr/lib/libobk.a
```

8.4.3.1 Environment Variables

The ADSMConnect Agent can use unique environment variables to locate files, allowing for API applications different files from those used by the interactive client.

These environment variables can be set in the Oracle user′s profile. However, when executed, RMAN does not fully inherit the environment from the Oracle user. All of the environment variables have default values, which RMAN will use. These defaults can be overridden by using RMAN options (see 8.4.3.3, “RMAN Parms Option” on page 176).

The following environment variables are required for the ADSMConnect Agent:

DSMI_CONFIG Fully qualified name that points to the dsm.opt file.

DSMI_DIR Fully qualified name that points to the directory that contains the dsm.sys file and the verification file

DSMI_LOG Fully qualified name that points to the directory that contains the API error log file (dsierror.log)

The following environment variables are optional:

DSMO_AVG_SIZE The average size of objects. This variable is passed to the ADSM server where it is used to determine storage pool and device usage. It is recommended that the DSMO_AVG_SIZE environment variable be set. In Version 2.1.0.6, the units are megabytes, with a default of 50 MB.

During normal ADSM client to ADSM server backup operations, the ADSM client notifies the ADSM server of the size of the next object that will be sent. The ADSM server determines whether space is available to store the object before allowing the object to be sent. With RMAN the server is not notified of the object size; rather it is given the value determined by the DSMO_AVG_SIZE environment variable. If this value is set lower than the actual object size, the ADSM server must halt transfer operations to determine whether additional space is available. This process can cause problems on the ADSM server. It is recommended that the DSMO_AVG_SIZE environment variable be set to a number equal to the size of the largest Oracle data file within the database. When you set the size of the variable, pay attention to threshold maximums set on ADSM server storage pools. For example, if a disk storage pool accepts files to a maximum of 20 MB and the value of DSMO_AVG_SIZE is set to 40000 (40 MB), the object would be forced to the next storage pool (usually tape).

DSM_FS The filespace name that will be created on the ADSM server. This environment variable provides you with control over the filespace name for the RMAN backup. In fact, you can use several different names to differentiate your backups. When setting up this option, do not use a / before the filesystem name. The default is /adsmorc.

> **Note**
>
> If you use many different filespace names, you must keep track of which name you use for each backup and ensure that you use the same name on the environment variable when you issue a restore.

DSMO_NODE The node name containing a string of 1 to 64 characters. This environment variable is used to specify a unique node name for the ADSMConnect Agent.

The default is the value returned by the AIX **hostname** command or the **NODENAME** value in the dsm.opt file.

DSMO_OWNER A session and object owner name containing a string of up to 64 characters. The default is a value returned by the **getuid** command. Use of this environment variable is not recommended.

DSMO_PSWDPATH Points to the directory where the ADSMO.yourhostname password file resides. The default is the current working directory.

As the ADSMConnect Agent is linked with Oracle, the execution environment for the ADSM API calls will be that of the environment provided for the Oracle backup process started with RMAN. Environment variables such as DSMI_DIR

and DSMO_FS, which can be specified in the Oracle user environment, are not included by the RMAN process environment because the RMAN process environment does not completely inherit the environment from the Oracle user′s environment (8.4.3.1, “Environment Variables” on page 173).

This is a problem if using the default directory for the ADSMConnect Agent. The ADSMConnect Agent is installed in /usr/lpp/adsmagent/aob, but the environment variables (most importantly DSMI_DIR, which points to the code) default to /usr/lpp/adsm/bin, preventing the ADSMConnect Agent from working. One solution is to move the ADSMConnect Agent to the default ADSM client directory. An alternative is to use RMAN options to supply environment variables.

8.4.3.2 Default Client Directory

As the ADSMConnect Agent does not receive environment variables from the RMAN process environment, it uses its default values for the variables unless they are overridden by using the RMAN parms option (8.4.3.3, “RMAN Parms Option” on page 176). For the ADSMConnect Agent to operate correctly the default client directory must contain the ADSMConnect Agent related files (Table 6 on page 170). This requirement is further complicated by the level of ADSMConnect Agent and backup-archive client installed:

- ADSMConnect Agent 2.1.0.6

  The libobk.a library provided with Version 2.1.0.6 of the ADSMConnect Agent has Version 2.1.0.6 of the ADSM API statically linked within it. The libobk.a library is linked into RMAN as part of the configuration process (8.4.3, “Linking ADSMConnect Agent to RMAN” on page 173) and is subsequently located by using the /usr/lib/libobk.a symbolic link. In addition to the libobk.a library, the dsm.sys and dsm.opt files and the dsmclient.cat message catalog must be located. The DSMI_DIR variable defaults to /usr/lpp/adsm/bin (backup-archive client directory), and the dsm.sys, and dsm.opt files in that directory and dsmclient.cat in the en_US subdirectory are used rather than the files created by the ADSMConnect Agent in the /usr/lpp/adsmagent/aob directory. The files used are part of the Version 2 backup-archive client.

  If Version 3 of the backup-archive client has been installed, the Version 3 message catalog, dsmclientv3.cat, will be in the /usr/lpp/adsm/bin/en_US subdirectory. In this situation the ADSMConnect Agent will not work because it cannot locate dsmclient.cat in the /usr/lpp/adsm/bin/en_US subdirectory. The solution is to copy the dsmclient.cat message catalog installed by the ADSMConnect Agent from the /usr/lpp/adsmagent/aob/en_US directory to the /usr/lpp/adsm/bin/en_US directory.

- ADSMConnect Agent 2.1.0.7

  The libobk.a library provided with Version 2.1.0.7 of the ADSMConnect Agent is dynamically linked to Version 3.1.3 of the ADSM API client, and the ADSM API client must be installed separately. The ADSM API library, libApiDS.a, must be accessible from the /usr/lib directory, which is done by creating a symbolic link from /usr/lib/libApiDS.a to /usr/lpp/adsm/api/libApiDS.a.

  The default installation directory for the 2.1.0.7 ADSMConnect Agent is /usr/lpp/adsmagent/aob. The libobk.a library in this directory is linked into RMAN and accessed in the same way as for Version 2.1.0.6 of the ADSMConnect Agent. The DSMI_DIR variable still defaults to /usr/lpp/adsm/bin (backup-archive client directory), and the dsm.sys and

dsm.opt in that directory are used rather than the files created by the ADSMConnect Agent in the /usr/lpp/adsmagent/aob directory. The Version 2.1.0.7 ADSMConnect Agent uses the Version 3 client message catalog, dsmclientv3.cat, which is in the /usr/lpp/adsm/bin/en_US directory as a result of installing the Version 3.1.3 API client.

8.4.3.3 RMAN Parms Option

RMAN requires that a channel be allocated before RMAN commands can be issued to perform backup and restore operations with ADSM. The Oracle backup process environment can be extended by using *parms* options with the allocate channel command:

```
% rman target internal/manager@DFM8 rcvcat cat8dba/cat8dba@CAT8
RMAN>
run {
allocate channel adsm type 'SBT_TAPE' parms
'ENV=(DSMI_DIR=/usr/lpp/adsmagent/aob,DSMO_FS=adsmorc)'
}
RMAN>
```

Oracle appends to a sbtio.log log file in the $ORACLE_HOME/rdbms/log directory. This log file contains information about code levels and return codes from both the Oracle SBT API and ADSM API calls. Using the ADSM API header file, dsmrc.h, included with the ADSM API code installation can assist in diagnosing ADSMConnect Agent problems.

8.5 Using RMAN Commands

This section is not a definitive guide to RMAN commands as only those commands used in the examples are covered. For further information about RMAN commands, refer to *Oracle8 Server Backup and Recovery Guide Release 8.0*. The following topics are covered:

- Invoking RMAN
- Channel control commands
- Backup command
- Restore command
- Recover command
- Maintenance commands

The native RMAN interface is command-line only. An additional Oracle product, the Oracle Enterprise Manager, can be used to provide a graphical interface to RMAN and EBU.

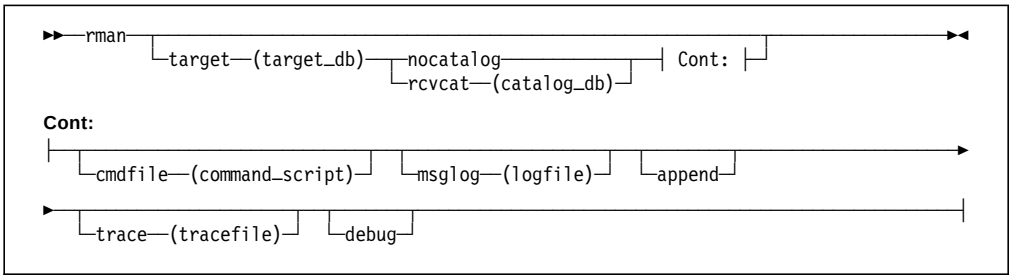
8.5.1 Invoking RMAN

The **rman** command is an AIX command provided with Oracle8 that invokes RMAN. The **rman** command supports both interactive and batch mode operations. At an AIX prompt, switching to a userid with the correct Oracle environment set, and then entering **rman** invokes RMAN in interactive mode and displays the RMAN prompt:

```
root@severn:/ # su - oracle

oracle@severnDFM8 %rman

Recovery Manager: Release 8.0.3.0.0 - Production

RMAN>
```

Individual commands used within RMAN can now be entered at this prompt. An alternative method of using RMAN is to run it in batch mode. Using this method the individual commands used within RMAN are defined in a script file and the script file is executed with the **rman** command.

Entering the **rman** command only starts the RMAN shell environment. It does not connect to any database instances or enable any functions to be performed. Typically RMAN must connect to the target and recovery catalog database before any other commands can be executed. To do this the **rman** command has a number of options and parameters for connecting to target and recovery catalog databases, running script files, and performing logging and tracing operations:



Option	Description
target	Specifies that RMAN will connect to a target database
target_db	The connection string for the target database. This takes the form of *userid/password@SID*.
nocatalog	Invokes RMAN without connecting to a recovery catalog database. This has implications on the type of backups and restoration possible. See 8.1.2, “RMAN System Components” on page 157 for more details.
rcvcat	Specifies that RMAN will connect to a catalog database
catalog_db	The connection string for the catalog database. This takes the form of *userid/password@SID*.
cmdfile	Executes a command script in batch mode
command_script	The path and name of the script that RMAN will execute
msglog	Logs the output of all commands executed to a file. If this option is not specified, no logs are created and RMAN writes out messages to stdout. If this option is specified, RMAN writes messages only to the logfile, not to stdout.
logfile	The path and name of the logfile that RMAN will create

append When used with **msglog**, the logfile will be appended to and not truncated.

trace Provides trace information for commands executed. This option is issued in conjunction with the **debug** option, which must also be specified to create output. Specifying trace without debug does not produce any output.

tracefile The path and name of the trace file that RMAN will create. If not specified with the debug option, debug output will be included in the log file.

debug Enables detailed debug information for subsequent commands. This information is written to the trace file if specified.

The following example invokes RMAN and connects to the DFM8 target database as the *internal* user and to the CAT8 recovery catalog as the *cat8dba* user:

```
%rman target internal/manager@DFM8 rcvcat cat8dba/cat8dba@CAT8

Recovery Manager: Release 8.0.3.0.0 - Production

RMAN-06005: connected to target database: DFM8
RMAN-06008: connected to recovery catalog database
RMAN>
```

After connecting to the target and recovery catalog instances, additional RMAN commands can be run from the RMAN> prompt. It is also possible to run RMAN without a recovery catalog by specifying the nocatalog option.

Executing RMAN displays all messages to the screen (stdout). No message logging takes place. The msglog option and a filename can be specified with the **rman** command to redirect messages to a log file:

```
% rman target internal/manager@DFM8 rcvcat cat8dba/cat8dba@CAT8 msglog "'rman.log'"
RMAN>
```

This invokes RMAN and redirects all output to the rman.log file. When the msglog option is used, no messages other than the RMAN> prompt is displayed on the screen. All messages are buffered and written to the rman.log only when RMAN is exited.

Arguments in Quoted Strings

Option arguments such as cmdfile, logfile, and tracefile must be specified as quoted (' ') strings. However, when entering arguments as quoted strings, shell processing of quotes characters must to be considered. Use the back slash (/) character or additional double quote (") characters around the quoted string to prevent quote character removal by the shell.

An alternative method of displaying messages on the screen and creating a log file is to use standard AIX facilities, such as the **tee** command. Executing **rman** and piping (|) the output to **tee** displays all messages on the screen and writes them to the specified log file:

```
%rman target internal/manager@DFM8 rcvcat cat8dba/cat8dba@CAT8 | tee -a rman.log

Recovery Manager: Release 8.0.3.0.0 - Production

RMAN-06005: connected to target database: DFM8
RMAN-06008: connected to recovery catalog database
RMAN>
```

The **tee - a** option appends to the specified log file rather than replacing it.

Once RMAN is invoked, additional commands can be entered at the RMAN> prompt. These commands, covered in the next sections, are used to perform backup and restore functions on the target databases. The first command that must be entered is **run**. It sets up a line mode input where additional commands can be entered:

```
RMAN> run
>2 {
>3 <command 1> ;
>4 <command 2> ;
>5 }
```

Once **run** has been entered, all subsequent commands must be entered within braces **{** and **}**. Multiple commands can be entered within the braces, with each command followed by a semi-colon (;), which denotes the end of that command. Catalog maintenance commands such as listing backups are executed outside of a **run** command, directly from the RMAN> prompt.

Native SQL commands can also be performed from either the RMAN> prompt or within a run command. These are performed by entering **sql**, the SQL command in quotes, followed by a semicolon. This function is used for tasks such as taking a tablespace offline before restoring and recovering it:

```
RMAN> sql 'alter tablespace users offline' ;

RMAN-03022: compiling command: sql
RMAN-06162: sql statement: alter tablespace users offline
RMAN-03023: executing command: sql
RMAN>
```

Command scripts held on a local filesystem can be used to define RMAN operations. A command script is created by placing the required RMAN commands, including the **run** command, braces, and semicolons, in a plain text file. This script can then be executed by using the **cmdfile** option with the script name as an argument:

```
% rman target internal/manager@DFM8 rcvcat cat8dba/cat8dba@CAT8 cmdfile "'script name'"
```

RMAN is exited by entering **exit** at the RMAN> prompt:

```
RMAN> exit

Recovery Manager complete.
%
```

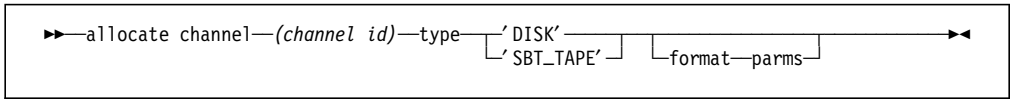
The sections that follow describe some of the commands that can be issued at the RMAN> prompt or within scripts to perform backup, restore, and recovery of target databases and other administrative functions.

8.5.2 Channel Control Commands

Channel control commands are used to connect RMAN to ADSM and enable backup, restore, and recover operations to be performed. This section covers commands for allocating channels.

8.5.2.1 Allocate Channel

The **allocate channel** command establishes a server process that handles the connection between the target database instance and the backup media, in this case, the ADSM server. At least one channel must be allocated before a backup, restore, or recover operation can be performed. Multiple channels can be allocated if required. The **allocate channel** command has the following syntax:



Option	Description
channel id	A name for the channel to be allocated. It can be any character string.
type	Defines the type of media to be used for subsequent functions. Specify **type** followed by '**DISK**' to use local disk files or '**SBT_TAPE**' to use tape, which in these examples is the ADSMConnect Agent. These device types must be entered in single quotes.
format	Defines the name format to use for the backup pieces that are created on this channel. This is optional and can be used if format is not specified on the backup command.
parms	Defines additional parameters for the device being allocated. It can be used to define external variables to RMAN such as ADSMConnect Agent environment variables.

In the following example a channel named ADSM is allocated with two ADSMConnect Agent environment variables set:

```
RMAN > run
2> {
3> allocate channel ADSM type 'SBT_TAPE'
4> parms 'ENV=(DSMI_DIR=/usr/lpp/adsmagent/aob,DSMO_FS=adsmorc)';
5> }

RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: adsm
RMAN-08500: channel adsm: sid=11 devtype=SBT_TAPE
RMAN-08031: released channel: adsm
RMAN>
```

The channel is allocated and then released. Channels are released automatically at the end of a string of commands defined by the end brace character (}).

Command Continuation

The allocate channel example above illustrates how commands can be continued on multiple lines. Pressing Enter while entering a command starts a new line where the command can be continued. Commands entered on multiple lines are completed with a semicolon character.

8.5.2.2 Allocate Channel for Delete

Channels used for backup, restore, or recovery purposes are allocated as the first command within a **run** command.

A channel can also be allocated for RMAN maintenance purposes such as deleting a backup from the recovery catalog and the ADSM server. This channel allocation must be done before the backup can be deleted. Backups to be deleted must have been created on the same device type being used for the delete. The allocate channel for delete is not issued within a **run** command:

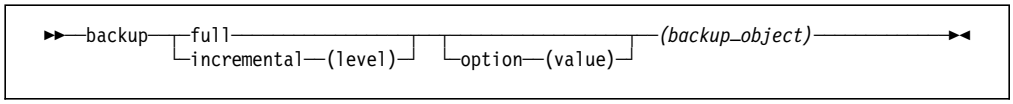
```
RMAN> allocate channel for delete type 'SBT_TAPE';

RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: delete
RMAN-08500: channel delete: sid=11 devtype=SBT_TAPE
```

This delete channel is not deleted automatically and must be deleted by using a **release channel** command:

```
RMAN> release channel ;

RMAN-03022: compiling command: release
RMAN-03023: executing command: release
RMAN-08031: released channel: delete
RMAN>
```

8.5.3 Backup Command

The **backup** command is used to create one or more backup sets. A backup set contains one or more data files, control files, or archive logs from a target database. The **backup** command is entered within a **run** command following an **allocate channel** command. It has the following syntax:



The type of backup, full or incremental, must be specified. Full is the default. If incremental is specified, an incremental backup level must also be specified. The **backup** command has a number of additional options, including:

Option	Description
cumulative	Specifies that the backup will be cumulative (see 8.2.1.3, “Incremental Backup” on page 160)
filesperset	Specifies the maximum number of files to place in one backup set. If the number of files backed up is greater than the fileperset, multiple backup sets will be created.
tag	A backup set can be given a user-specified identifier called a **tag**. Tag characters must be 30 characters or less and typically contain descriptive text such as “Daily Incremental Backup.”
format	Specifies the file name to use for the backup pieces. The name must be enclosed in quotes. The format operand can be specified in the **backup** or **allocate channel** commands. The following substitution variables are available for the format string: **%s** The backup set number **%p** The backup piece number within the backup set **%d** The database name **%n** The padded database name **%t** The backup set stamp name **%u** A short form consisting of backup set number and the time the backup set was created The name created by the format option will be the name of the backup object stored on the ADSM server.

Finally a *(backup_object)* must also be specified. The objects that can be backed up include the entire database, tablespaces, data files, and control files.

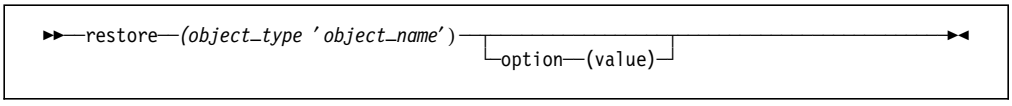
The following is an example of a level 0 incremental backup of the users tablespace:

```
RMAN> run
2> {
3> allocate channel ADSM type 'SBT_TAPE' ;
4> backup incremental level 0 tag Level0_Users_Table filesperset 10
4> format '%d_%s_%p'
5> (tablespace users) ;
6> }

RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: adsm
RMAN-08500: channel adsm: sid=11 devtype=SBT_TAPE

RMAN-03022: compiling command: backup
RMAN-03023: executing command: backup
RMAN-08008: channel adsm: starting datafile backupset
  ⋮
```

The backup is given a tag of Level0_Users_Table, a maximum of 10 files is included in each backup set, and the name of the object stored on the ADSM server is determined from resolving the %d_%s_%p format option to DFM8_28_1. DFM8 is the database ID, 28 is the number of the backup set, and it is backup piece 1 in that set.

8.5.4 Restore Command

The **restore** command is used to restore full and incremental level 0 backup sets. The whole database, tablespaces, or individual data files can be restored. The **restore** command has the following syntax:



The **restore** command has the following options:

Option	Description
restore object	The restore object contains the list of objects to be restored. This can be the database, tablespaces, data files, control files, or archive logs. If individual data files are being restored, the object name must also be in quotes.
from	Specifies whether a restore should restore from file copies or from backup sets. The default is the most recent backup set.
tag	Used to filter backup sets for restores based on the tag values used during backups.

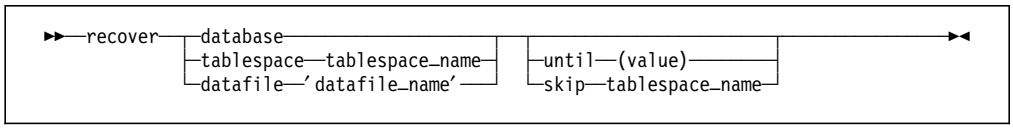
Multiple files within individual restore operations are restored in parallel if multiple channels are allocated. The following example restores the previously backed up user's tablespace. Before the restore can take place, the tablespace must be taken offline with an SQL **alter tablespace** command:

```
RMAN> run
2> {
3> allocate channel ADSM type 'SBT_TAPE' ;
4> sql 'alter tablespace users offline immediate' ;
5> restore (tablespace users) ;
6> sql 'alter tablespace users online' ;
7> }
  ⋮
```

This example will produce errors when the **alter tablespace users online** command is processed because the restored tablespace must be recovered, by using the **recover** command, before it is put online.

8.5.5 Recover Command

The **recover** command is used to recover a database, tablespace, or data file to a consistent state. It is run after a restore operation. It performs media recovery and applies any appropriate incremental backups. It has the following syntax:



Option	Description
database	The entire database is to be recovered.
tablespace	A list of one or more tablespaces to recover
datafile	A list of one or more data files to recover
skip	Used with database recovery only. This statement will skip a tablespace that should not be recovered. RMAN will take offline the data files belonging to these tablespaces.
until	Used to specify a point-in-time recovery

In the following example the users tablespace is restored and recovered while the database in online:

```
RMAN> run {
2> allocate channel ADSM type 'SBT_TAPE' ;
3> sql 'alter tablespace users offline immediate' ;
4> restore (tablespace users) ;
5> recover tablespace users ;
6> sql 'alter tablespace users online' ;
7> }
```

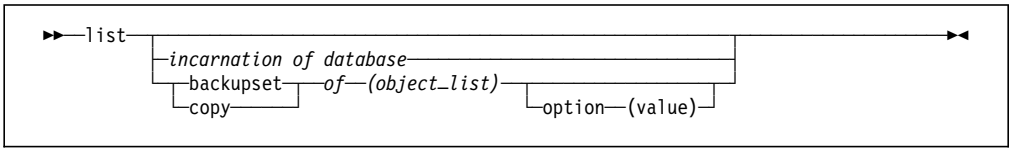
This is the same example as for the **restore** command, but recovery is being performed before the tablespace is put back online.

8.5.6 Maintenance Commands

In addition to the channel, backup, restore, and recovery commands, there are maintenance commands used to perform maintenance operations on the recovery catalog. This section covers the list, report, change, and stored script commands.

8.5.6.1 List

The **list** command is used to query the contents of the recovery catalog. It can be used to display target databases registered with the recovery catalog and to determine which backups are available for the database. It has the following syntax:



Option	Description
incarnation of database	Lists all target database instances registered with the recovery catalog
backup type	The backup type to be listed, either backup set or copy
(object_list)	Specifies the object to be listed. The object can be a database, tablespace, or data file.
until time option	Used to list all backups made before the specified time
from time option	Used to list all backups made since the specified time
device type option	Used to list all backup sets that were made to the specified device type.

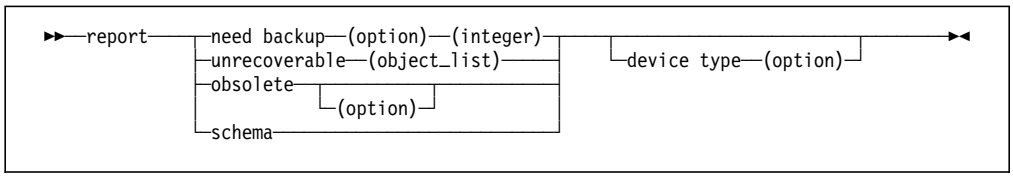
The following example lists all backups of the user tablespace on device 'SBT_TAPE' that were made before 06 Mar 1998:

```
RMAN> list backupset of tablespace users
2> device type 'SBT_TAPE'
3> until time '06 mar 1998' ;

RMAN-03022: compiling command: list
RMAN-06230: List of Datafile Backups
RMAN-06231: Key     File Type         LV Completion_time Ckp SCN    Ckp Time
RMAN-06232: ------- ---- ------------ -- --------------- ---------- --------
RMAN-06233: 6102    5    Full            27-FEB-98       216989     27-FEB-98
RMAN-06233: 6124    5    Full            27-FEB-98       217010     27-FEB-98
RMAN-06233: 6141    5    Full            02-MAR-98       217040     02-MAR-98
RMAN-06233: 6160    5    Full            02-MAR-98       217058     02-MAR-98
RMAN>
```

The **list** command can be executed from the RMAN> prompt. It does not go with a run statement.

8.5.6.2 Report

The **report** command can be used to issue more complex queries on the recovery catalog. It has the following syntax:



The following reports can be produced:

Option	Description
report need backup	Lists all data files that are considered to need a new backup. This report requires that days or incremental be specified as an option followed by an integer number: • days *n*. Reports the data files that have not had a backup for more than *n* days • incremental *level*. Reports the data files that require a backup because *x* or more incremental backups were taken since the last level 0 backup.
unrecoverable	Lists all the data files that are unrecoverable. The report can include the entire database, tablespaces, or data files.
obsolete	Lists all backup sets and backup pieces that are obsolete
schema	Reports on the database schema

The following example is a report on all data files that need a new backup because three or more incremental backups have been taken since the last level 0 backup:

```
RMAN> report need backup incremental 3 database;
```

8.5.6.3 Change

The **change** command is used to change the state of an object within the recovery catalog. You can change the state for all kinds of backup pieces.

Option	Description
uncatalog	Removes reference of a backup piece or backup set from the recovery catalog
delete	Same as uncatalog, but in addition, media management is called to delete the backup piece or backup set. This command must be preceded by an allocate channel for delete command, specifying the appropriate device type.
validate	Removes reference of a backup piece or backup set from the recovery catalog when that file no longer exists on the disk
until time	Changes the status of all backups made previous to the specified date

The following example deletes a backup piece that was stored by media management from the recovery catalog:

```
RMAN> allocate channel for delete type 'sbt_tape';
RMAN> change backuppiece <primary key> delete;
RMAN> release channel;
```

8.5.6.4 Stored Scripts

RMAN command scripts can be stored in the recovery catalog and executed by using the RMAN **execute script** command. Storing scripts in the recovery catalog instead of text file scripts in the file system has the advantage that they can be used on multiple target databases and maintained centrally by an administrator.

The **create script** command is used to create a stored script:

```
RMAN> create script db_full
1> {
2> allocate channel ADSM type 'SBT_TAPE' ;
3> backup full format '%d_%s_%p' (database) ;
4> }
```

The **execute script** command within a run command is used to execute the stored script:

```
RMAN>run {
2> execute script db_full;
}
```

The **replace script** command can be used to replace the current stored script. The **delete script** deletes a stored script. The **print script** prints the script to the screen or to a log file.

8.6 Backup and Recovery Examples

This section describes some typical backup and recovery examples using RMAN, complete with sample scripts. The examples are based on the following database instances:

- Target database(Oracle 8.0.3) - DFM8
- Recovery catalog database(Oracle 8.0.3) - CAT8

For the examples we used a simplified method to invoke RMAN without having to enter the connect string each time it was run. We created two aliases for the **rman** command by adding the following lines to the Oracle user's C shell profile (.cshrc):

```
alias rmancmd "rman target internal/manager@DFM8 rcvcat cat8dba/cat8dba@CAT8  cmdfile"
alias rmanl   "rman target internal/manager@DFM8 rcvcat cat8dba/cat8dba@CAT8"
```

These alias definitions enable **rman** to be executed with target and recovery catalog connect strings by running a single command, the alias. The **rmancmd** alias starts RMAN, connects to the target and recovery catalog instances, and executes a stored command file that must be entered as a parameter:

```
% rmancmd cmdfile_name
```

The command file is a regular AIX text file that contains the commands executed by RMAN.

The **rmanl** alias starts RMAN, connects to the target and recovery catalog instances, and displays the RMAN> prompt.

We then created an RMAN stored script that allocates a channel to ADSM. Using the **rmanl** alias, we started RMAN and created the script called *alloc_adsm*, using the **create script** command:

```
%rmanl

Recovery Manager: Release 8.0.3.0.0 - Production

RMAN-06005: connected to target database: DFM8
RMAN-06008: connected to recovery catalog database
RMAN> create script alloc_adsm
2> {
3> allocate channel ADSM type 'SBT_TAPE' ;
4> }

RMAN-03022: compiling command: create script
RMAN-03023: executing command: create script
RMAN-08085: created script alloc_adsm
RMAN>
```

This script, stored in the recovery catalog, simplifies the process of allocating channels at the start of RMAN jobs. It can be used from any target database that connects to the recovery catalog. It is executed by using the **execute script** command as the first command within an RMAN run statement:

```
RMAN> run
2> {
3> execute script alloc_adsm ;
   ⋮
```

The examples illustrate how to perform the following functions:

1. Data file backup
2. Tablespace incremental backup
3. Database backup
4. Archived redo logs backup
5. Data file restore
6. Tablespace restore
7. Database restore
8. Database point-in-time restore
9. Backup and restore using multiple channels

These examples use command scripts stored on local filesystems, which in turn execute a combination of RMAN stored scripts and RMAN commands. To illustrate their purpose, they are executed from an AIX command prompt. For automation purposes they can be executed by using the AIX scheduler cron or the ADSM backup-archive client scheduler.

8.6.1 Data File Backup

This example shows how to back up an individual data file with RMAN. A tablespace consists of one or more data files. The tablespace called test_ts01 consists of two data files. We show how to back up a single data file while the database is online and open.

Figure 73 is an example of a stored script called dfback that backs up a single data file.

```
# script name: dfback
#
# task:  Perform single datafile backup
#
run {
execute script alloc_adsm ;                                          1
backup (datafile '/u01/oradata/DFM8/test02.dbf' format '%d_%s_%p');  2
}
```

Figure 73. RMAN Data File Backup Script

The dfbacks script contains an RMAN run statement with two commands within the braces:

1 An **execute script** command that runs the previously created and stored alloc_adsm script to allocate a channel to ADSM

2 A **backup** command that backs up the test02.dbf data file using a $d_$s_$d format string.

The script is created by the Oracle user and stored in the /home/oracle/rman/scripts directory. It is then executed by using the **rmancmd** alias with the script path and name as a parameter with the output redirected to a dfback.log file:

```
% rmancmd /home/oracle/rman/scripts/dfback | tee dfback.log
```

Figure 74 shows the contents of dfback.log after the job has executed.

```
Recovery Manager: Release 8.0.3.0.0 - Production

RMAN-06005: connected to target database: DFM8
RMAN-06008: connected to recovery catalog database
RMAN> # script name:  dfback
2> #
3> # task: Perform single datafile backup
4> #
5>
6> run {
7> execute script alloc_adsm;
8> backup
9> (datafile '/u01/oradata/DFM8/test02.dbf' format '%d_%s_%p');
10> }

11>
RMAN-03021: executing script: alloc_adsm
```

Figure 74 (Part 1 of 2). Log of RMAN Data File Backup

```
RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: adsm
RMAN-08500: channel adsm: sid=8 devtype=SBT_TAPE                3

RMAN-03022: compiling command: backup
RMAN-03023: executing command: backup
RMAN-08008: channel adsm: starting datafile backupset
RMAN-08502: set_count=161 set_stamp=325170525
RMAN-08010: channel adsm: including datafile 7 in backupset
RMAN-08013: channel adsm: piece 1 created
RMAN-08503: piece handle=DFM8_161_1 comment=API
Version 122.156,MMS Version 32.42.163.216                       4
RMAN-03023: executing command: partial resync
RMAN-08003: starting partial resync of recovery catalog         5
RMAN-08005: partial resync complete
RMAN-08031: released channel: adsm                              6

Recovery Manager complete.
```

Figure 74 (Part 2 of 2). Log of RMAN Data File Backup

The first part of the log is a listing of the script that will be executed. The second part consists of messages that provide information about the various stages of execution:

3 The backup process is started by allocating a channel to ADSM.

4 The backup continues and a backup set consisting of one backup piece is created and sent to the ADSM server. The backup piece is given a handle, or name, of DFM8_161_1. This is the name of the object stored on the ADSM server.

5 After the backup set has been sent to ADSM, RMAN resynchronizes the target database with the recovery catalog. This step is necessary after every online backup operation because the control files and online redo log files may have been changed during the backup.

6 The last step is to release the channel to ADSM.

This completes the backup of the data file. This example shows a single data file being backed up. Multiple data files could be backed up by including additional data file names in the backup command.

8.6.2 Tablespace Incremental Backup

This example shows how to perform an incremental backup of a tablespace with RMAN. Two incremental backups, a level 0 and a level 2, are run for the test_ts01 tablespace. The tablespace used for this example consists of two data files.

8.6.2.1 Level 0 Backup

A level 0 incremental backup is a full backup of the tablespace. Figure 75 on page 191 is an example of a stored script called tsinc0 that performs the level 0 backup.

```
# script name:  tsinc0
#
# tasks this script executes level 0 incremental backup
# for test_ts01 table space
#
run {
execute script alloc_adsm;
backup
 incremental level 0                              1
 format '%d_%s_%p'
 tag level_0_TEST_TS01 (tablespace test_ts01);    2
}
```

Figure 75. RMAN Tablespace Level 0 Incremental Backup Script

The tsinc0 script contains an RMAN run statement with two commands within it. The first command is the **execute script** command that allocates a channel to ADSM. The second command is a **backup** command with the following parameters:

1 The *incremental level 0* option specifies the type of backup to be performed.

2 The tag identifier adds a description of level_0_TEST_TS01 to the backup of the test_ts01 tablespace.

The script is stored in the /home/oracle/rman/scripts directory. It is executed by using the **rmancmd** alias with the output redirected to a tsinc0.log file:

```
% rmancmd /home/oracle/rman/scripts/tsinc0 | tee tsinc0.log
```

Figure 76 shows the contents of tsinc0.log after the backup operation.

```
Recovery Manager: Release 8.0.3.0.0 - Production

RMAN-06005: connected to target database: DFM8
RMAN-06008: connected to recovery catalog database
RMAN> # script name: tsinc0
2> #
3> # tasks  this script executes level 0 incremental backup
4> # for test_ts01 table space
5> #
6> run {
7> execute script alloc_adsm;
8> backup
9>  incremental level 0
10>  format '%d_%s_%p'
11>  tag level_0_TEST_TS01
12>  (tablespace test_ts01);
13> }
   :
RMAN-03022: compiling command: backup
RMAN-03023: executing command: backup
RMAN-08008: channel adsm: starting datafile backupset
RMAN-08502: set_count=164 set_stamp=325171712
RMAN-08010: channel adsm: including datafile 6 in backupset     3
RMAN-08010: channel adsm: including datafile 7 in backupset
RMAN-08013: channel adsm: piece 1 created
RMAN-08503: piece handle=DFM8_164_1 comment=API
Version 122.156,MMS Version 32.42.163.216                       4
```

Figure 76 (Part 1 of 2). RMAN Log of Tablespace Level 0 Backup

```
RMAN-03023: executing command: partial resync
RMAN-08003: starting partial resync of recovery catalog          5
RMAN-08005: partial resync complete
RMAN-08031: released channel: adsm

Recovery Manager complete.
```

Figure 76 (Part 2 of 2). RMAN Log of Tablespace Level 0 Backup

3 The backup includes both data files (6 and 7) within the tablespace in the backup set.

4 The backup piece sent to ADSM is named DFM8_164_1.

5 As this tablespace backup is an online backup, RMAN resynchronizes the recovery catalog with the target database.

The final step is to release the allocated channel.

8.6.2.2 Level 2 Backup

This example shows the level 2 backup of the tablespace. After the level 0 backup, updates are made to the tablespace in order to have some changed data blocks to back up with the level 2 incremental backup.

After the update operation (not shown), the level 2 backup of the tablespace is performed. Figure 77 is an example of a stored script named tsinc2 that performs the level 2 incremental backup.

```
# script name:  tsinc2
#
# tasks this script executes level 2 incremental backup
# for test_ts01 table space
#
run {
execute script alloc_adsm;
backup
 incremental level 2              1
 format '/test_ts01/%d_%s_%p'     2
 tag level_2_TEST_TS01
 (tablespace TEST_TS01)
 (current controlfile);           3
}
```

Figure 77. RMAN Tablespace Level 2 Incremental Backup Script

The tsinc2 script contains an RMAN run statement with two commands within it. The first command is the **execute script** command that allocates a channel to ADSM. The second command is a **backup** command with the following parameters:

1 The *incremental level 2* option specifies the type of backup to be performed.

2 For the level 2 backup, a different name for the backup is used by using a different format string. The backup piece is named /test_ts01/DFM8_167_1 to better distinguish between the level 0 and level 2 backup objects.

3 The current control file is backed up. This is shown to illustrate the use of additional parameters of the **backup** command.

The script is stored in the /home/oracle/rman/scripts directory. It is executed by using the **rmancmd** alias with the output redirected to a tsinc2.log file:

```
% rmancmd /home/oracle/rman/scripts/tsinc2 | tee tsinc2.log
```

Figure 78 shows the contents of tsinc2.log after the backup operation.

```
Recovery Manager: Release 8.0.3.0.0 - Production

RMAN-06005: connected to target database: DFM8
RMAN-06008: connected to recovery catalog database
RMAN> # script name: tsinc2
   :
RMAN-03023: executing command: backup
RMAN-08008: channel adsm: starting datafile backupset
RMAN-08502: set_count=167 set_stamp=325173748
RMAN-08010: channel adsm: including datafile 6 in backupset
RMAN-08010: channel adsm: including datafile 7 in backupset
RMAN-08013: channel adsm: piece 1 created
RMAN-08503: piece handle=/test_ts01/DFM8_167_1 comment=API                    4
Version 122.156,MMS Version 32.42.163.216
RMAN-08008: channel adsm: starting datafile backupset
RMAN-08502: set_count=168 set_stamp=325173765
RMAN-08011: channel adsm: including current controlfile in backupset     5
RMAN-08013: channel adsm: piece 1 created
RMAN-08503: piece handle=/test_ts01/DFM8_168_1 comment=API
Version 122.156,MMS Version 32.42.163.216                                      6
RMAN-03023: executing command: partial resync
RMAN-08003: starting partial resync of recovery catalog
RMAN-08005: partial resync complete
RMAN-08031: released channel: adsm

Recovery Manager complete.
```

Figure 78. RMAN Log of Tablespace Level 2 Backup

4 Any data files containing blocks that have changed since the level 0 backup are backed up and sent to ADSM as a backup piece named /test_ts01/DFM8_167_1.

5 The current control file is backed up separately when the data file backup is complete.

6 The current control file is sent to ADSM as a separate backup piece, /test_ts01/DFM8_168_1, within the backup set for this level 2 backup.

The final step is to resynchronize the recovery catalog with the target database and to release the previously allocated channel.

8.6.3 Database Backup

This example shows how to perform a whole database backup, that is, a backup of the complete database that includes all tablespaces and the control files.

Figure 79 on page 194 is an example of a stored script called dbback that performs a whole database backup operation.

```
# script name: dbback
#
# task   whole database backup
# including the control files
#
run {
execute script alloc_adsm;
backup
 full                    1
 tag whole_DFM8_backup
 filesperset 2           2
 format 'whole_%d_%s'
 (database);             3
}
```

Figure 79. Whole Database Backup Script

The dbback script contains an RMAN run statement with two commands within it. The first command is the **execute script** command that allocates a channel to ADSM. The second command is a **backup** command with the following parameters:

1 The *full* parameter is specified to perform a whole database backup.

2 A *filesperset 2* parameter is also specified to put a maximum of two data files in each backup set.

3 The whole database is backed up.

The script is stored in the /home/oracle/rman/scripts directory. It is executed by using the **rmancmd** alias with the output redirected to a dbback.log file:

```
% rmancmd /home/oracle/rman/scripts/dbback | tee dbback.log
```

Figure 80 shows the contents of dbback.log following the backup operation.

```
Recovery Manager: Release 8.0.3.0.0 - Production

RMAN-06005: connected to target database: DFM8
RMAN-06008: connected to recovery catalog database
RMAN> # script name: dbback
2> #
3> # task   whole database backup
4> # including the control files
5> #
6>
7> run {
8> execute script alloc_adsm;
9> backup
10>  full
11>  tag whole_DFM8_backup
12>  filesperset 2
13>  format 'whole_%d_%s'
14>  (database);
15> }
   ⋮
RMAN-03023: executing command: backup
RMAN-08008: channel adsm: starting datafile backupset
RMAN-08502: set_count=169 set_stamp=325173852
```

Figure 80 (Part 1 of 2). RMAN Log of Whole Database Backup

```
RMAN-08010: channel adsm: including datafile 1 in backupset          4
RMAN-08010: channel adsm: including datafile 2 in backupset
RMAN-08013: channel adsm: piece 1 created
RMAN-08503: piece handle=whole_DFM8_169 comment=API
Version 122.156,MMS Version 32.42.163.216
RMAN-08008: channel adsm: starting datafile backupset
RMAN-08502: set_count=170 set_stamp=325173952
RMAN-08010: channel adsm: including datafile 3 in backupset
RMAN-08010: channel adsm: including datafile 4 in backupset
RMAN-08013: channel adsm: piece 1 created
RMAN-08503: piece handle=whole_DFM8_170 comment=API
Version 122.156,MMS Version 32.42.163.216
RMAN-08008: channel adsm: starting datafile backupset
RMAN-08502: set_count=171 set_stamp=325173981
RMAN-08010: channel adsm: including datafile 5 in backupset
RMAN-08010: channel adsm: including datafile 6 in backupset
RMAN-08013: channel adsm: piece 1 created
RMAN-08503: piece handle=whole_DFM8_171 comment=API
Version 122.156,MMS Version 32.42.163.216
RMAN-08008: channel adsm: starting datafile backupset
RMAN-08502: set_count=172 set_stamp=325173999
RMAN-08010: channel adsm: including datafile 7 in backupset          5
RMAN-08011: channel adsm: including current controlfile in backupset 6
RMAN-08013: channel adsm: piece 1 created
RMAN-08503: piece handle=whole_DFM8_172 comment=API
Version 122.156,MMS Version 32.42.163.216
RMAN-03023: executing command: partial resync
RMAN-08003: starting partial resync of recovery catalog
RMAN-08005: partial resync complete
RMAN-08031: released channel: adsm

Recovery Manager complete.
```

Figure 80 (Part 2 of 2). RMAN Log of Whole Database Backup

4 The first two data files within the database are sent to ADSM.

5 The database consists of seven data files. These are sent to ADSM two at a time. This results in four backup sets being sent to ADSM, with the fourth containing a single data file.

6 The current control file is also backed up and sent as a separate backup set to ADSM. For a whole database backup operation, there is no need to specify the *current controlfile* parameter. The control file is automatically backed up.

The final step is to resynchronize the recovery catalog with the target database and to release the previously allocated channel.

8.6.4 Archived Redo Log Backup

This example shows how to back up the archived redo logs. Before you back up the archived redo logs, Oracle recommends that you archive the current online redo log file by using the **alter system archive current log** command.

Figure 81 on page 196 is an example of a stored script called arclog that archives the current online redo log file, allocates a channel to ADSM, and then backs up all the archived redo logs.

```
# script name:  arclog
#
# tasks    archive the current redo log
#          backup the archive logs, keep them on the disk
#
run {
sql 'alter system archive log current';  1
execute script alloc_adsm;
backup
  filesperset 2
  format 'archive_log_%d_%s'
  (archivelog all);                      2
}
```

Figure 81. RMAN Archived Redo Log Backup Script

The arclog script contains an RMAN run statement with three commands within it:

1 The first command is the SQL *alter system archive log current* command. This archives the current redo log. The second command allocates a channel to ADSM.

2 The final command backs up all the archived redo logs.

The script is stored in the /home/oracle/rman/scripts directory. It is executed by using the **rmancmd** alias with the output redirected to a arclog.log file:

```
% rmancmd /home/oracle/rman/scripts/arclog | tee arclog.log
```

Figure 82 shows the contents of arclog.log following the backup operation.

```
Recovery Manager: Release 8.0.3.0.0 - Production

RMAN-06005: connected to target database: DFM8
RMAN-06008: connected to recovery catalog database
RMAN> # script name:  arclog
2> #
3> # tasks    archive the current redo log
4> #          backup the archive logs, keep them on the disk
5> #
6> run {
7> sql 'alter system archive log current';
8> execute script  alloc_adsm;
9> backup
10>   filesperset 2
11>   format 'archive_log_%d_%s'
12>   (archivelog all);
13> }

RMAN-03022: compiling command: sql
RMAN-06162: sql statement: alter system archive log current     3
RMAN-03023: executing command: sql
RMAN-03021: executing script: alloc_adsm

RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: adsm
RMAN-08500: channel adsm: sid=8 devtype=SBT_TAPE
```

Figure 82 (Part 1 of 2). RMAN Log of Archived Log Backup

```
RMAN-03022: compiling command: backup
RMAN-03025: performing implicit partial resync of recovery catalog                    4
RMAN-03023: executing command: partial resync
RMAN-08003: starting partial resync of recovery catalog
RMAN-08005: partial resync complete
RMAN-03023: executing command: backup
RMAN-08009: channel adsm: starting archivelog backupset
RMAN-08502: set_count=36 set_stamp=328114009
RMAN-08014: channel adsm: including archivelog in backup set                          5
RMAN-08504: input archivelog thread=1 sequence=1 recid=3 stamp=327513784
RMAN-08014: channel adsm: including archivelog in backup set
RMAN-08504: input archivelog thread=1 sequence=2 recid=4 stamp=328112902
RMAN-08013: channel adsm: piece 1 created
RMAN-08503: piece handle=archive_log_DFM8_36 comment=API Version 0.1,MMS Version 2.1.6.0 6
RMAN-08009: channel adsm: starting archivelog backupset
RMAN-08502: set_count=37 set_stamp=328114105
RMAN-08014: channel adsm: including archivelog in backup set
RMAN-08504: input archivelog thread=1 sequence=3 recid=5 stamp=328112979
RMAN-08014: channel adsm: including archivelog in backup set
RMAN-08504: input archivelog thread=1 sequence=4 recid=6 stamp=328113655
RMAN-08013: channel adsm: piece 1 created
RMAN-08503: piece handle=archive_log_DFM8_37 comment=API Version 0.1,MMS Version 2.1.6.0
RMAN-08009: channel adsm: starting archivelog backupset
RMAN-08502: set_count=38 set_stamp=328114113
RMAN-08014: channel adsm: including archivelog in backup set
RMAN-08504: input archivelog thread=1 sequence=5 recid=7 stamp=328113923
RMAN-08014: channel adsm: including archivelog in backup set
RMAN-08504: input archivelog thread=1 sequence=6 recid=8 stamp=328114002               7
RMAN-08013: channel adsm: piece 1 created
RMAN-08503: piece handle=archive_log_DFM8_38 comment=API Version 0.1,MMS Version 2.1.6.0
RMAN-03023: executing command: partial resync
RMAN-08003: starting partial resync of recovery catalog
RMAN-08005: partial resync complete
RMAN-08031: released channel: adsm

Recovery Manager complete.
```

Figure 82 (Part 2 of 2). RMAN Log of Archived Log Backup

3 The *alter system archive log current* SQL command is processed.

4 After the *alter system archive log current* command, RMAN performs a partial resynchronization of the recovery catalog because the target database control file has been updated.

5 RMAN sends the archived redo logs to ADSM, using two files per backupset. Each of the archived redo logs has a unique sequence number.

6 RMAN uses a backuppiece name of *archive_log_DFM8_36* to identify the backuppiece containing the first two archived logs.

7 This process is repeated for the remaining archived redo logs. Each log has a unique sequence number, and each backuppiece created is also given a unique name.

The final step is to resynchronize the recovery catalog and target database again and to release the allocated channel.

8.6.5 Data File Restore

This example shows how to restore a single data file. This task has to be performed whenever a data file is lost or damaged. After the restore is complete, a recovery must be run to make the tablespace consistent. As part of the recover operation, the archive redo logs are also restored. This example restores the test02.dbf data file backed up as part of the whole database backup in 8.6.3, “Database Backup” on page 193.

Figure 83 on page 198 is an example of a stored script called restdf that restores and recovers a single datafile.

```
# script name:  restdf
#
# tasks         restore a single datafile within a tablespace
#
run {
execute script alloc_adsm;
sql 'alter tablespace test_ts01 offline immediate';  1
restore (datafile '/u01/oradata/DFM8/test02.dbf');   2
recover tablespace test_ts01;                        3
sql 'alter tablespace test_ts01 online';             4
}
```

Figure 83. RMAN Restore Data File Script

The restdf script contains the stored script for allocating a channel to ADSM and an RMAN run statement with four commands:

1 An *alter tablespace test_ts01 offline immediate* SQL command is issued to vary offline the tablespace that contains the data file being recovered.

2 A **restore** command for the data file

3 A **recover** command that recovers the tablespace to a consistent state

4 An *alter tablespace test_ts01 online* SQL command is issued to vary online the tablespace following the recovery.

The script is stored in the /home/oracle/rman/scripts directory. It is executed by using the **rmancmd** alias with the output redirected to a restdf.log file:

```
% rmancmd /home/oracle/rman/scripts/restdf | tee restdf.log
```

Figure 84 shows the contents of restdf.log after the restore and recovery operation.

```
Recovery Manager: Release 8.0.3.0.0 - Production

RMAN-06005: connected to target database: DFM8
RMAN-06008: connected to recovery catalog database
RMAN> # script name:  restdf
   ⋮
RMAN-03021: executing script: alloc_adsm

RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: adsm
RMAN-08500: channel adsm: sid=14 devtype=SBT_TAPE

RMAN-03022: compiling command: sql
RMAN-06162: sql statement: alter tablespace test_ts01 offline immediate    5
RMAN-03023: executing command: sql

RMAN-03022: compiling command: restore
RMAN-03023: executing command: restore
```

Figure 84 (Part 1 of 2). RMAN Log for Restored and Recovered Data File

```
RMAN-08016: channel adsm: starting datafile backupset restore              6
RMAN-08502: set_count=172 set_stamp=325173999
RMAN-08019: channel adsm: restoring datafile 7
RMAN-08509: destination for restore of datafile 7: /u01/oradata/DFM8/test02.dbf
RMAN-08023: channel adsm: restored backup piece 1                          7
RMAN-08511: piece handle=whole_DFM8_172 params=NULL
RMAN-08024: channel adsm: restore complete
RMAN-03023: executing command: partial resync                              8
RMAN-08003: starting partial resync of recovery catalog
RMAN-08005: partial resync complete

RMAN-03022: compiling command: recover                                     9
RMAN-03022: compiling command: recover(1)
RMAN-03022: compiling command: recover(2)
RMAN-03022: compiling command: recover(3)
RMAN-03023: executing command: recover(3)
RMAN-08054: starting media recovery                                       10
RMAN-08515: archivelog filename=/u01/app/oracle/product/803/dbs/arch1_480.dbf thread=1
sequence=480
RMAN-08515: archivelog filename=/u01/app/oracle/product/803/dbs/arch1_481.dbf thread=1
sequence=481
RMAN-08055: media recovery complete

RMAN-03022: compiling command: recover(4)
RMAN-03023: executing command: partial resync
RMAN-08003: starting partial resync of recovery catalog
RMAN-08005: partial resync complete

RMAN-03022: compiling command: sql
RMAN-06162: sql statement: alter tablespace test_ts01 online              11
RMAN-03023: executing command: sql
RMAN-08031: released channel: adsm

Recovery Manager complete.
```

Figure 84 (Part 2 of 2). RMAN Log for Restored and Recovered Data File

5 The *alter tablespace test_ts01 offline immediate* SQL statement is compiled and run to take the tablespace offline.

6 The data file restore operation is started. Information about the restore backup set such as the backup set number is obtained from the recovery catalog.

7 The handle of the backup piece being restored is displayed.

8 The recovery catalog is resynchronized before the start of the recovery operation.

9 The recover command is compiled to determine which recovery operations must be performed and which archived redo logs will be required.

10 Media recovery is performed by using archived redo logs restored from ADSM.

11 The *alter tablespace test_ts01 online* SQL statement is compiled and run to put the tablespace back online at completion of the restore and recovery operation.

The final step is to release the previously allocated channel.

8.6.6 Tablespace Restore

This example shows how to restore a tablespace. The tablespace being restored contains two data files and was backed up with a combination of incremental level 0 and 2 backups. The restore operation is followed by a recovery operation.

Figure 85 is an example of a stored script called resttb that restores and recovers the tablespace.

```
#  script name:   resttb
#
#  task          restore tablespace to the most current state
#
run {
execute script alloc_adsm;
sql 'alter tablespace test_ts01 offline immediate';   1
restore (tablespace test_ts01);                       2
recover tablespace test_ts01;                         3
sql 'alter tablespace test_ts01 online';              4
}
```

Figure 85. RMAN Restore Tablespace Script

The resttb script contains the stored script for allocating a channel to ADSM and an RMAN run statement with four commands:

1. An *alter tablespace test_ts01 offline immediate* SQL command is issued to vary offline the tablespace that is being restored.
2. A **restore** command for the tablespace
3. A **recover** command that recovers the tablespace to a consistent state
4. An *alter tablespace test_ts01 online* SQL command is issued to vary online the tablespace after the recovery.

The script is stored in the /home/oracle/rman/scripts directory. It is executed by using the **rmancmd** alias with the output redirected to a resttb.log file:

```
% rmancmd /home/oracle/rman/scripts/resttb | tee resttb.log
```

Figure 86 shows the contents of resttb.log after the restore and recovery operation.

```
Recovery Manager: Release 8.0.3.0.0 - Production

RMAN-06005: connected to target database: DFM8
RMAN-06008: connected to recovery catalog database
   ⋮
RMAN-03022: compiling command: sql
RMAN-06162: sql statement: alter tablespace test_ts01 offline immediate
RMAN-03023: executing command: sql
RMAN-03022: compiling command: restore
RMAN-03025: performing implicit partial resync of recovery catalog
RMAN-03023: executing command: partial resync
```

Figure 86 (Part 1 of 2). RMAN Log for Restored and Recovered Tablespace

```
RMAN-08003: starting partial resync of recovery catalog
RMAN-08005: partial resync complete
RMAN-03023: executing command: restore                                    5
RMAN-08016: channel adsm: starting datafile backupset restore
RMAN-08502: set_count=186 set_stamp=325182215
RMAN-08019: channel adsm: restoring datafile 6                            6
RMAN-08509: destination for restore of datafile 6: /u01/app/oracle/product/803/dbs/test01.dbf
RMAN-08019: channel adsm: restoring datafile 7
RMAN-08509: destination for restore of datafile 7: /u01/oradata/DFM8/test02.dbf
RMAN-08023: channel adsm: restored backup piece 1                         7
RMAN-08511: piece handle=DFM8_186_1 params=NULL
RMAN-08024: channel adsm: restore complete
RMAN-03023: executing command: partial resync
RMAN-08003: starting partial resync of recovery catalog
RMAN-08005: partial resync complete
RMAN-03022: compiling command: recover
RMAN-03022: compiling command: recover(1)
RMAN-03022: compiling command: recover(2)

RMAN-03023: executing command: recover(1)
RMAN-08039: channel adsm: starting incremental datafile backupset restore  8
RMAN-08502: set_count=187 set_stamp=325182386
RMAN-08019: channel adsm: restoring datafile 6
RMAN-08509: destination for restore of datafile 6: /u01/app/oracle/product/803/dbs/test01.dbf
RMAN-08019: channel adsm: restoring datafile 7
RMAN-08509: destination for restore of datafile 7: /u01/oradata/DFM8/test02.dbf
RMAN-08023: channel adsm: restored backup piece 1
RMAN-08511: piece handle=/test_ts01/DFM8_187_1 params=NULL
RMAN-08024: channel adsm: restore complete                                9

RMAN-08039: channel adsm: starting incremental datafile backupset restore
RMAN-08502: set_count=189 set_stamp=325182602
RMAN-08019: channel adsm: restoring datafile 6
RMAN-08509: destination for restore of datafile 6:
/u01/app/oracle/product/803/dbs/test01.dbf
RMAN-08019: channel adsm: restoring datafile 7
RMAN-08509: destination for restore of datafile 7:
/u01/oradata/DFM8/test02.dbf
RMAN-08023: channel adsm: restored backup piece 1
RMAN-08511: piece handle=/test_ts01/DFM8_189_1 params=NULL
RMAN-08024: channel adsm: restore complete

RMAN-08039: channel adsm: starting incremental datafile backupset restore
RMAN-08502: set_count=191 set_stamp=325182820
RMAN-08019: channel adsm: restoring datafile 6
RMAN-08509: destination for restore of datafile 6:
/u01/app/oracle/product/803/dbs/test01.dbf
RMAN-08019: channel adsm: restoring datafile 7
RMAN-08509: destination for restore of datafile 7:
/u01/oradata/DFM8/test02.dbf
RMAN-08023: channel adsm: restored backup piece 1
RMAN-08511: piece handle=/test_ts01/DFM8_191_1 params=NULL
RMAN-08024: channel adsm: restore complete
  ⋮
Recovery Manager complete.
```

Figure 86 (Part 2 of 2). RMAN Log for Restored and Recovered Tablespace

5 The restore operation is started by using the channel allocated to ADSM. Only the level 0 backup is restored.

6 The data files needed for the restore are listed, including the destination for the files when restored.

7 The backup piece is restored, followed by a partial resynchronization of the recovery catalog and target database.

8 The recovery process starts by restoring and applying the incremental backups. Each required incremental backup is restored separately.

9 The first incremental backup restore is complete. This restore process then runs again for the two additional incremental backups that are required to complete the recovery.

The final steps (not shown) are resynchronizing the recovery catalog, varying the tablespace online, and releasing the allocated channel.

8.6.7 Database Restore

This example shows how to restore a database to its most current state. The database must be started but not mounted. The database instance can be started in nomount mode by using svrmgrl:

```
SVRMGR> startup nomount
ORACLE instance started.
Total System Global Area       5536980 bytes
Fixed Size                       42592 bytes
Variable Size                  4150900 bytes
Database Buffers                819200 bytes
Redo Buffers                    524288 bytes
```

This command starts the instance, mounts the database, but does not open it.

After the restore is complete, a recover must be performed to make the database consistent. The database must be mounted before the recovery process.

Figure 87 is an example of a stored script called restdb that restores and recovers a database to its most current state.

```
# script name:  restdb
#
# task        restore the database to its most current state
#
#
run {
execute script alloc_adsm;
restore database;                 1
sql 'alter database mount';       2
recover database;                 3
}
```

Figure 87. RMAN Restore Database Script

The restdb script contains the stored script for allocating a channel to ADSM and an RMAN run statement with three commands:

1 A **restore** command for the database

2 An **alter database mount** SQL command is issued to mount the database before the recovery.

3 A **recover** command that recovers the database to its most current state

The script is stored in the /home/oracle/rman/scripts directory. It is executed by using the **rmancmd** alias with the output redirected to a restdb.log file:

```
% rmancmd /home/oracle/rman/scripts/restdb | tee restdb.log
```

Figure 88 on page 203 shows the contents of restdb.log after the restore and recovery operation.

```
Recovery Manager: Release 8.0.3.0.0 - Production

RMAN-06006: connected to target database: DFM8 (not mounted)
RMAN-06008: connected to recovery catalog database
   ⋮
RMAN-03022: compiling command: restore
RMAN-03023: executing command: restore
RMAN-08016: channel adsm: starting datafile backupset restore   4
RMAN-08502: set_count=169 set_stamp=325173852
RMAN-08019: channel adsm: restoring datafile 1
RMAN-08509: destination for restore of datafile 1: /u01/oradata/DFM8/system01.dbf
RMAN-08019: channel adsm: restoring datafile 2
RMAN-08509: destination for restore of datafile 2: /u01/oradata/DFM8/rbs01.dbf
RMAN-08023: channel adsm: restored backup piece 1
RMAN-08511: piece handle=whole_DFM8_169 params=NULL
RMAN-08024: channel adsm: restore complete
   ⋮
RMAN-03022: compiling command: sql
RMAN-06162: sql statement: alter database mount                 5
RMAN-03023: executing command: sql
   ⋮
RMAN-03022: compiling command: recover                          6
RMAN-03022: compiling command: recover(1)
RMAN-03023: executing command: recover(1)
RMAN-08039: channel adsm: starting incremental datafile backupset restore
RMAN-08502: set_count=187 set_stamp=325182386
RMAN-08019: channel adsm: restoring datafile 6
RMAN-08509: destination for restore of datafile 6: /u01/app/oracle/product/803/dbs/test01.dbf
RMAN-08023: channel adsm: restored backup piece 1
RMAN-08511: piece handle=/test_ts01/DFM8_187_1 params=NULL
RMAN-08024: channel adsm: restore complete
   ⋮
RMAN-08024: channel adsm: restore complete

RMAN-03023: executing command: recover(3)
RMAN-08054: starting media recovery                             7
RMAN-08515: archivelog filename=/u01/app/oracle/product/803/dbs/arch1_480.dbf
thread=1 sequence=480
RMAN-08515: archivelog filename=/u01/app/oracle/product/803/dbs/arch1_481.dbf
thread=1 sequence=481
RMAN-08515: archivelog filename=/u01/app/oracle/product/803/dbs/arch1_482.dbf
thread=1 sequence=482
RMAN-08055: media recovery complete
RMAN-08031: released channel: adsm
Recovery Manager complete.
```

Figure 88. RMAN Log for Restored and Recovered Database

4 The restore operation is started. All the backup sets for the most current whole or incremental level 0 backup are restored.

5 After the restore an *alter database mount* SQL command is issued to mount the newly restored database.

6 The recover process restores any incremental backups that were taken subsequent to the whole or incremental level 0 backup that has been restored.

7 The media recovery process starts, which restores the archive redo logs and rolls forward the recovered database to its most current state.

The final step is releasing the allocated channel.

The database can now be opened by using svrmgrl:

```
SVRMGR> alter database open ;
Statement processed.
SVRMGR>
```

8.6.8 Database Point-in-Time Restore

This example shows how to restore a database to a point in time. It is a similar process to 8.6.7, “Database Restore” on page 202 except that a recovery is not required after a point-in-time restore operation. When doing a point-in-time restore the database must be started in nomount mode (as in 8.6.7, “Database Restore” on page 202).

Figure 89 is an example of a stored script named restdbpit that performs a point-in-time restore.

```
# script:  restdbpit
#
# task   database point in time restore
#
run {
execute script alloc_adsm;
restore database until time '14 feb 1998'; 1
sql 'alter database mount'; 2
}
```

Figure 89. RMAN Database Point-in-Time Restore Script

The *until time* option is one method of specifying a point of recovery. Other options are system change numbers (SCNs) and log sequence numbers.

The restdbpit script contains the stored script for allocating a channel to ADSM and an RMAN run statement with two commands:

1 A **restore database** command with an *until time* parameter specifying the point in time to which the database should be restored.

2 An *alter database mount* SQL command is issued to mount the database after the recovery.

The script is stored in the /home/oracle/rman/scripts directory. It is executed by using the **rmancmd** alias with the output redirected to a restdbpit.log file:

```
% rmancmd /home/oracle/rman/scripts/restdbpit | tee restdbpit.log
```

Figure 90 shows the contents of restdbpit.log after the restore and recovery operation.

```
Recovery Manager: Release 8.0.3.0.0 - Production

RMAN-06006: connected to target database: DFM8 (not mounted)
RMAN-06008: connected to recovery catalog database
RMAN> # script name restdbpit
```

Figure 90 (Part 1 of 2). RMAN Log of Database Point-in-Time Restore

```
   ⋮
RMAN-03023: executing command: restore                          3
RMAN-08016: channel dev1: starting datafile backupset restore
RMAN-08502: set_count=169 set_stamp=325173852
RMAN-08019: channel dev1: restoring datafile 1
RMAN-08509: destination for restore of datafile 1:
/u01/oradata/DFM8/system01.dbf
RMAN-08019: channel dev1: restoring datafile 2
RMAN-08509: destination for restore of datafile 2:
/u01/oradata/DFM8/rbs01.dbf
RMAN-08023: channel dev1: restored backup piece 1
RMAN-08511: piece handle=whole_DFM8_169 params=NULL
RMAN-08024: channel dev1: restore complete                      4
   ⋮
RMAN-08019: channel dev1: restoring datafile 7
RMAN-08509: destination for restore of datafile 7: /u01/oradata/DFM8/test02.dbf
RMAN-08023: channel dev1: restored backup piece 1
RMAN-08511: piece handle=DFM8_186_1 params=NULL
RMAN-08024: channel dev1: restore complete
RMAN-03022: compiling command: sql                              5
RMAN-06162: sql statement: alter database mount
RMAN-03023: executing command: sql
RMAN-08031: released channel: dev1

Recovery Manager complete.
```

Figure 90 (Part 2 of 2). RMAN Log of Database Point-in-Time Restore

3 The restore operation is started. All the backup sets for the most current whole or incremental level 0 backup are restored.

4 The incremental level 2 backups required to recover to the specified point in time are restored.

5 After the restore an *alter database mount* SQL command is issued to mount the newly restored database.

In this example the database was restored to a date consistent with a level 2 incremental backup. The database can now be opened. Opening the database may result in a message that recovery is required. Recovery can be performed by running **recover** from svrmgrl:

```
SVRMGR> alter database open;
alter database open
*
ORA-01113: file 1 needs media recovery
ORA-01110: data file 1: '/u01/oradata/DFM8/system01.dbf'
SVRMGR> recover
Media recovery complete.
SVRMGR>
SVRMGR> alter database open;
Statement processed.
SVRMGR>
```

In the above example archived redo logs were not available for recovery of data file 1, so recovery just reset the data file. If archived redo logs had been available, the log names would have been listed, and they could be selected for recovery.

8.6.9 Backup and Restore Using Multiple Channels

All of the previous examples have used a single channel to ADSM. To improve backup and restore performance for larger databases, multiple, parallel channels can be used to multiplex the backup and restore data streams.

This example shows how to back up and restore a database by using multiple channels. The backup job performs an incremental level 0 backup of the whole database.

Figure 91 is an example of a stored script named muxback that performs a multiplexed database backup.

```
#script name  muxback
#
# tasks     this script execute level 0 incremental backup
#           for whole database using 7 channels
#
run {
allocate channel adsm1 type 'SBT_TAPE';            1
allocate channel adsm2 type 'SBT_TAPE';
allocate channel adsm3 type 'SBT_TAPE';
allocate channel adsm4 type 'SBT_TAPE';
allocate channel adsm5 type 'SBT_TAPE';
allocate channel adsm6 type 'SBT_TAPE';
allocate channel adsm7 type 'SBT_TAPE';
backup                                             2
    incremental level 0
    filesperset 1
    Format '%d_%s_%p/Mux_1'
    tag Level_0_DB
    (database) ;
}
```

Figure 91. RMAN Database Multiplexed Backup Script

The muxback script contains seven allocate channel commands **1**. Seven were allocated because this database consists of seven physical files that will be backed up: six data files and one control file. After the allocate channel commands is a backup command **2** that performs an incremental level 0 backup of the whole database, using the filesperset option with a value of 1. This option creates a separate backup set for each data file and the control file. As seven channels have been allocated, each backup set will be sent to ADSM on individual channels.

The script is stored in the /home/oracle/rman/scripts directory. It is executed by using the **rmancmd** alias with the output redirected to a muxback.log file:

```
% rmancmd /home/oracle/rman/scripts/muxback | tee muxback.log
```

Figure 92 on page 207 shows the contents of muxback.log after the backup operation.

```
Recovery Manager: Release 8.0.3.0.0 - Production

RMAN-06005: connected to target database: DFM8
RMAN-06008: connected to recovery catalog database
RMAN> #script name  muxback
2> #
3> # tasks     this script execute level 0 incremental backup
4> #           for whole database using 7 channels
5> #
6> run {
7> allocate channel adsm1 type 'SBT_TAPE';
   ⋮
13> allocate channel adsm7 type 'SBT_TAPE';
14> backup
15>   incremental level 0
16>   filesperset 1
17>   Format '%d_%s_%p/Mux_1'
18>   tag Level_0_DB
19>   (database) ;
20> }

21>

RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: adsm1
RMAN-08500: channel adsm1: sid=15 devtype=SBT_TAPE              3
   ⋮
RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: adsm7
RMAN-08500: channel adsm7: sid=17 devtype=SBT_TAPE

RMAN-03022: compiling command: backup
RMAN-03023: executing command: backup
RMAN-08008: channel adsm3: starting datafile backupset          4
RMAN-08502: set_count=109 set_stamp=328791464
RMAN-08008: channel adsm5: starting datafile backupset
RMAN-08502: set_count=111 set_stamp=328791464
RMAN-08008: channel adsm6: starting datafile backupset
RMAN-08502: set_count=112 set_stamp=328791464
RMAN-08008: channel adsm7: starting datafile backupset
RMAN-08502: set_count=113 set_stamp=328791464
RMAN-08008: channel adsm1: starting datafile backupset
RMAN-08502: set_count=107 set_stamp=328791464
RMAN-08008: channel adsm2: starting datafile backupset
RMAN-08502: set_count=108 set_stamp=328791464
RMAN-08010: channel adsm3: including datafile 3 in backupset
RMAN-08008: channel adsm4: starting datafile backupset
RMAN-08502: set_count=110 set_stamp=328791464
RMAN-08010: channel adsm5: including datafile 5 in backupset
RMAN-08010: channel adsm6: including datafile 6 in backupset
RMAN-08010: channel adsm1: including datafile 1 in backupset
RMAN-08010: channel adsm2: including datafile 2 in backupset
RMAN-08010: channel adsm4: including datafile 4 in backupset
RMAN-08011: channel adsm7: including current controlfile in backupset
RMAN-08013: channel adsm3: piece 1 created
RMAN-08503: piece handle=DFM8_109_1/Mux_1 comment=API Version 0.1,MMS Version 2.1.6.0   4
RMAN-08013: channel adsm5: piece 1 created
RMAN-08503: piece handle=DFM8_111_1/Mux_1 comment=API Version 0.1,MMS Version 2.1.6.0
RMAN-08013: channel adsm7: piece 1 created
RMAN-08503: piece handle=DFM8_113_1/Mux_1 comment=API Version 0.1,MMS Version 2.1.6.0
RMAN-08013: channel adsm4: piece 1 created
RMAN-08503: piece handle=DFM8_110_1/Mux_1 comment=API Version 0.1,MMS Version 2.1.6.0
RMAN-08013: channel adsm6: piece 1 created
RMAN-08503: piece handle=DFM8_112_1/Mux_1 comment=API Version 0.1,MMS Version 2.1.6.0
```

Figure 92 (Part 1 of 2). RMAN Log of Database Multiplexed Backup

```
RMAN-08013: channel adsm1: piece 1 created
RMAN-08503: piece handle=DFM8_1O7_1/Mux_1 comment=API Version 0.1,MMS Version 2.1.6.0
RMAN-08013: channel adsm2: piece 1 created
RMAN-08503: piece handle=DFM8_1O8_1/Mux_1 comment=API Version 0.1,MMS Version 2.1.6.0
RMAN-03023: executing command: partial resync
RMAN-08003: starting partial resync of recovery catalog
RMAN-08005: partial resync complete
RMAN-08031: released channel: adsm2
RMAN-08031: released channel: adsm1
RMAN-08031: released channel: adsm4
RMAN-08031: released channel: adsm6
RMAN-08031: released channel: adsm3
RMAN-08031: released channel: adsm5
RMAN-08031: released channel: adsm7

Recovery Manager complete.
```

Figure 92 (Part 2 of 2). RMAN Log of Database Multiplexed Backup

The muxback.log shows that multiple channels have been allocated and the details of which files have been sent on which channels:

3 Seven channels, adsm1 to adsm7, are allocated.

4 The backup job starts. Seven backup sets are created, each containing one of the data files or the control file.

5 Each of the backup pieces is sent to ADSM on separate channels.

This example results in seven parallel sessions to ADSM with database files in a separate backup set and stored as separate objects on the ADSM server. Different combinations of number of channels and fileperset values can be used. If filesperset 3 had been used, only three channels would have been used (three files on the first session, three files on the second session, and one file on the third session) even though seven channels had been allocated. The number of channels to be allocated is limited by the capacity of the network and the ADSM server, in particular, the number of tape drives available if the backups are going directly to tape.

The restore process is similar to 8.6.7, “Database Restore” on page 202. First the database must be started in nomount mode:

```
SVRMGR> startup nomount ;
ORACLE instance started.
Total System Global Area       5536980 bytes
Fixed Size                       42592 bytes
Variable Size                  4150900 bytes
Database Buffers                819200 bytes
Redo Buffers                    524288 bytes
SVRMGR>
```

Then the database can be restored and recovered by using multiple channels. Figure 93 on page 209 is an example of a stored script named muxrest that performs a multiplexed database restore.

```
#script name  muxrest
#
# tasks       this script restores database
#             using multiple channels
#
run {
allocate channel adsm1 type 'SBT_TAPE';          1
allocate channel adsm2 type 'SBT_TAPE';
allocate channel adsm3 type 'SBT_TAPE';
allocate channel adsm4 type 'SBT_TAPE';
allocate channel adsm5 type 'SBT_TAPE';
allocate channel adsm6 type 'SBT_TAPE';
allocate channel adsm7 type 'SBT_TAPE';
restore database;                                2
sql 'alter database mount';
recover database;
}
```

Figure 93. RMAN Database Multiplexed Restore Script

As with the muxback script, this script allocates seven channels to ADSM **1**. The subsequent commands **2** are similar to the previous database restore example (8.6.7, "Database Restore" on page 202). They include a database restore, an alter database mount SQL command, and a recover database command.

The script is stored in the /home/oracle/rman/scripts directory. It is executed by using the **rmancmd** alias with the output redirected to a muxrest.log file:

```
% rmancmd /home/oracle/rman/scripts/muxrest | tee muxrest.log
```

Figure 94 shows the contents of muxrest.log after the restore operation.

```
Recovery Manager: Release 8.0.3.0.0 - Production

RMAN-06006: connected to target database: DFM8 (not mounted)
RMAN-06008: connected to recovery catalog database
RMAN> #script name  muxrest
2> #
3> # tasks       this script restores database
4> #             using multiple channels
5> #
6> run {
7> allocate channel adsm1 type 'SBT_TAPE';
   ⋮
13> allocate channel adsm7 type 'SBT_TAPE';
14> restore database;
15> sql 'alter database mount';
16> recover database;
17> }

18>

RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: adsm1
```

Figure 94 (Part 1 of 3). RMAN Log of Database Multiplexed Restore

```
RMAN-08500: channel adsm1: sid=11 devtype=SBT_TAPE                3
   ⋮
RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: adsm7
RMAN-08500: channel adsm7: sid=17 devtype=SBT_TAPE

RMAN-03022: compiling command: restore
RMAN-03023: executing command: restore
RMAN-08016: channel adsm6: starting datafile backupset restore
RMAN-08502: set_count=112 set_stamp=328791464
RMAN-08016: channel adsm1: starting datafile backupset restore
RMAN-08502: set_count=107 set_stamp=328791464
RMAN-08019: channel adsm1: restoring datafile 1                                    4
RMAN-08509: destination for restore of datafile 1: /u01/oradata/DFM8/system01.dbf
RMAN-08016: channel adsm2: starting datafile backupset restore
RMAN-08502: set_count=108 set_stamp=328791464
RMAN-08019: channel adsm2: restoring datafile 2
RMAN-08509: destination for restore of datafile 2: /u01/oradata/DFM8/rbs01.dbf
RMAN-08016: channel adsm3: starting datafile backupset restore
RMAN-08502: set_count=109 set_stamp=328791464
RMAN-08019: channel adsm3: restoring datafile 3
RMAN-08509: destination for restore of datafile 3: /u01/oradata/DFM8/temp01.dbf
RMAN-08016: channel adsm4: starting datafile backupset restore
RMAN-08502: set_count=110 set_stamp=328791464
RMAN-08019: channel adsm4: restoring datafile 4
RMAN-08509: destination for restore of datafile 4: /u01/oradata/DFM8/tools01.dbf
RMAN-08016: channel adsm5: starting datafile backupset restore
RMAN-08502: set_count=111 set_stamp=328791464
RMAN-08019: channel adsm5: restoring datafile 5
RMAN-08509: destination for restore of datafile 5: /u01/oradata/DFM8/users01.dbf
RMAN-08019: channel adsm6: restoring datafile 6
RMAN-08509: destination for restore of datafile 6: /u01/oradata/DFM8/test_tim.dbf
RMAN-08023: channel adsm3: restored backup piece 1
RMAN-08511: piece handle=DFM8_109_1/Mux_1 params=NULL
RMAN-08024: channel adsm3: restore complete
RMAN-08023: channel adsm5: restored backup piece 1
RMAN-08511: piece handle=DFM8_111_1/Mux_1 params=NULL
RMAN-08024: channel adsm5: restore complete
RMAN-08023: channel adsm4: restored backup piece 1
RMAN-08511: piece handle=DFM8_110_1/Mux_1 params=NULL
RMAN-08024: channel adsm4: restore complete
RMAN-08023: channel adsm6: restored backup piece 1
RMAN-08511: piece handle=DFM8_112_1/Mux_1 params=NULL
RMAN-08024: channel adsm6: restore complete
RMAN-08023: channel adsm1: restored backup piece 1
RMAN-08511: piece handle=DFM8_107_1/Mux_1 params=NULL
RMAN-08024: channel adsm1: restore complete
RMAN-08023: channel adsm2: restored backup piece 1
RMAN-08511: piece handle=DFM8_108_1/Mux_1 params=NULL
RMAN-08024: channel adsm2: restore complete

RMAN-03022: compiling command: sql
RMAN-06162: sql statement: alter database mount                   5
RMAN-03023: executing command: sql

RMAN-03022: compiling command: recover

RMAN-03022: compiling command: recover(1)

RMAN-03022: compiling command: recover(2)

RMAN-03022: compiling command: recover(3)
RMAN-03023: executing command: recover(3)
RMAN-08054: starting media recovery
RMAN-08055: media recovery complete
```

Figure 94 (Part 2 of 3). RMAN Log of Database Multiplexed Restore

```
RMAN-03022: compiling command: recover(4)
RMAN-08031: released channel: adsm1
RMAN-08031: released channel: adsm2
RMAN-08031: released channel: adsm3
RMAN-08031: released channel: adsm4
RMAN-08031: released channel: adsm5
RMAN-08031: released channel: adsm6
RMAN-08031: released channel: adsm7

Recovery Manager complete.
```

Figure 94 (Part 3 of 3). RMAN Log of Database Multiplexed Restore

The muxrest.log shows the flow of the restore and recover operation:

3 Seven channels are allocated for the restore process.

4 Individual data files and the control file are restored on separate channels.

5 After the restore, the alter database mount SQL command is executed, and the recovery process run.

After the restore and recover, the database can be opened:

```
SVRMGR> alter database open;
Statement processed.
SVRMGR>
```

8.7 RMAN Administration

This section shows some examples of using RMAN commands for administrative purposes. The following examples are shown:

- Report target database schema
- List registered target databases
- List database backup sets
- Deleting obsolete backups
- Scheduling backups

These examples are produced by using the various RMAN maintenance commands introduced in 8.5.6, “Maintenance Commands” on page 185.

8.7.1 Report Target Database Schema

This example shows how to report the schema of the target database. It provides a simple overview of the target database structure.

This report is produced by using the **report** command with the *schema* option:

```
RMAN> report schema ;
```

Figure 95 on page 212 shows the output of the report schema command.

```
Recovery Manager: Release 8.0.3.0.0 - Production

RMAN-06005: connected to target database: DFM8
RMAN-06008: connected to recovery catalog database
RMAN>
RMAN-03022: compiling command: report
RMAN-06290: Report of database schema
RMAN-06291: File K-bytes    Tablespace        RB segs Name
RMAN-06292: ---- ---------- ----------------- ------- ------------------
RMAN-06293: 1         81920 SYSTEM            YES     /u01/oradata/DFM8/system01.dbf
RMAN-06293: 2         15360 RBS               YES     /u01/oradata/DFM8/rbs01.dbf
RMAN-06293: 3           552 TEMP              NO      /u01/oradata/DFM8/temp01.dbf
RMAN-06293: 4         25600 TOOLS             NO      /u01/oradata/DFM8/tools01.dbf
RMAN-06293: 5          1024 USERS             NO      /u01/oradata/DFM8/users01.dbf
RMAN-06293: 6          1024 TEST_TS01         NO      /u01/oradata/DFM8/test01.dbf
RMAN>       1
Recovery Manager complete.
```

Figure 95. RMAN Report Schema Command Output

All data files of a database get a unique data file number **1** within the recovery catalog. RMAN identifies the data files by this number.

8.7.2 List Registered Target Databases

The recovery catalog can be a shared resource for multiple target databases. These target databases are known as *incarnations* to the recovery catalog. The RMAN **list** command can be used to list the incarnations registered with the recovery catalog:

```
RMAN> list incarnation of database;
```

Figure 96 shows the output of the list incarnation command:

```
Recovery Manager: Release 8.0.3.0.0 - Production

RMAN-06005: connected to target database: DFM8
RMAN-06008: connected to recovery catalog database
RMAN>
RMAN-03022: compiling command: list
RMAN-06240: List of Database Incarnations
RMAN-06241: DB Key  Inc Key DB Name  DB ID            CUR Reset SCN  Reset Time
RMAN-06242: ------- ------- -------- ---------------- --- ---------- -------
RMAN-06243: 1       2726    DFM8     444689505        YES 216961     17-FEB-98
RMAN-06243: 662     663     MDG8     1672828043       YES 1          06-FEB-98
RMAN>
Recovery Manager complete.
```

Figure 96. RMAN List Incarnation Command Output

In the recovery catalog, two database instances, DFM8 and MDG8, are registered.

8.7.3 List Database Backup Sets

Over time as regular backups are performed an inventory of backup sets will be accumulated. These backup sets can be listed by using the **list** command with the *backupset* parameter:

```
RMAN> list backupset of database ;
```

The command produces a listing of all backup sets by data file, showing the data file numbers, the type of backup, and the date performed (Figure 97 on page 213).

```
RMAN-03022: compiling command: list
RMAN-06230: List of Datafile Backups
RMAN-06231: Key     File Type         LV Completion_time Ckp SCN    Ckp Time
RMAN-06232: ------- ---- ------------ -- --------------- ---------- --------
RMAN-06233: 6283    1    Incremental  0  17-MAR-98       217150     17-MAR-98
RMAN-06233: 6305    1    Incremental  2  17-MAR-98       217179     17-MAR-98
RMAN-06233: 6326    1    Incremental  2  17-MAR-98       217208     17-MAR-98
RMAN-06233: 6283    2    Incremental  0  17-MAR-98       217150     17-MAR-98
RMAN-06233: 6305    2    Incremental  2  17-MAR-98       217179     17-MAR-98
RMAN-06233: 6326    2    Incremental  2  17-MAR-98       217208     17-MAR-98
RMAN-06233: 6283    3    Incremental  0  17-MAR-98       217150     17-MAR-98
RMAN-06233: 6305    3    Incremental  2  17-MAR-98       217179     17-MAR-98
RMAN-06233: 6326    3    Incremental  2  17-MAR-98       217208     17-MAR-98
RMAN-06233: 6283    4    Incremental  0  17-MAR-98       217150     17-MAR-98
RMAN-06233: 6305    4    Incremental  2  17-MAR-98       217179     17-MAR-98
RMAN-06233: 6326    4    Incremental  2  17-MAR-98       217208     17-MAR-98
RMAN-06233: 6283    5    Incremental  0  17-MAR-98       217150     17-MAR-98
RMAN-06233: 6305    5    Incremental  2  17-MAR-98       217179     17-MAR-98
RMAN-06233: 6326    5    Incremental  2  17-MAR-98       217208     17-MAR-98
RMAN-06233: 6283    6    Incremental  0  17-MAR-98       217150     17-MAR-98
RMAN-06233: 6305    6    Incremental  2  17-MAR-98       217179     17-MAR-98
RMAN-06233: 6326    6    Incremental  2  17-MAR-98       217208     17-MAR-98
RMAN>
Recovery Manager complete.
```

Figure 97. RMAN List Backupsets Command Output

The File column lists the individual data file numbers within the database with a separate row for each backup set that contains a backup for that data file. The example was taken after one level 0 and two level 2 incremental backups for each data file.

8.7.4 Deleting Obsolete Backups

Over time as backup cycles become established, old and obsolete backups will accumulate. RMAN performs no automatic deletion of these backups. They must be manually deleted by using RMAN commands.

This example shows how to report obsolete backups and to remove them from the recovery catalog and the ADSM server. Two steps in this example fully illustrate the process:

1. Reporting the obsolete backups
2. Deleting the obsolete backups

8.7.4.1 Reporting Obsolete Backup Sets

RMAN keeps track of backup sets and backup pieces that have been superseded and therefore have become obsolete. These backup sets can be displayed by using the **report** command with the *obsolete* parameter:

```
RMAN> report obsolete;
```

Figure 98 on page 214 shows the output of the report obsolete command:

```
RMAN-03022: compiling command: report
RMAN-06280: Report of obsolete backup sets and datafile copies
RMAN-06281: Type                 Recid  Stamp     Filename
RMAN-06282: -------------------- ------ --------- -----------------------
RMAN-06284: Backup Set           33     328200301
RMAN-06285: Backup Piece         33     328200298 DFM8_42_1/Level_2
RMAN-06284: Backup Set           31     328198781
RMAN-06285: Backup Piece         31     328198778 DFM8_40_1/Level_0
RMAN>
```

Figure 98. RMAN Report Obsolete Command Output

The command lists the backup sets and pieces that are obsolete. The backup piece filename is the name of the object stored on the ADSM server.

This backup piece can also be viewed on the ADSM server by using an ADSM administrative command line client and issuing a **select** command to list the backup objects owned by the ADSMConnect Agent. Figure 99 illustrates a select command that lists the backup object state, name (ll_name), and backup date.

```
adsm> select state, ll_name, backup_date from backups where node_name='SEVERN'
  ⋮
             STATE     LL_NAME                    BACKUP_DATE
------------------     ------------------     ------------------
    ACTIVE_VERSION     DFM8_39_1/Level_0             1998-03-17
                                                14:36:18.000000
    ACTIVE_VERSION     DFM8_40_1/Level_0             1998-03-17
                                                14:37:50.000000
    ACTIVE_VERSION     DFM8_41_1/Level_2             1998-03-17
                                                15:01:45.000000
    ACTIVE_VERSION     DFM8_42_1/Level_2             1998-03-17
                                                15:03:11.000000
    ACTIVE_VERSION     DFM8_43_1/Level_2             1998-03-17
                                                15:15:37.000000
    ACTIVE_VERSION     DFM8_44_1/Level_2             1998-03-17
                                                15:17:06.000000
```

Figure 99. Displaying Active RMAN Objects on ADSM

The **DFM8_40_1/Level_0** backup piece, while obsolete to RMAN, is still an active backup version on the ADSM server. As an active backup version it will be held indefinitely by the ADSM server until deleted by RMAN.

ADSM Select Command

The administrative select command was introduced with ADSM Version 3. It is not available with Version 2 servers.

8.7.4.2 Deleting an Obsolete Backup Piece

An obsolete backup piece can be deleted by using the **change** command with the *backuppiece* parameter and the obsolete object filename. Figure 100 on page 215 illustrates such a command. The first command issued is **allocate channel for delete** followed by the **change backuppiece** command.

```
RMAN> allocate channel for delete type 'SBT_TAPE' ;
   ⋮
RMAN> change backuppiece 'DFM8_40_1/Level_0' delete ;

RMAN-03022: compiling command: change
RMAN-03023: executing command: change
RMAN-08073: deleted backup piece
RMAN-08517: backup piece handle=DFM8_40_1/Level_0 recid=31 stamp=328198778
RMAN-03023: executing command: partial resync
RMAN-08003: starting partial resync of recovery catalog
RMAN-08005: partial resync complete
RMAN>
```

Figure 100. Report of Files That Are Obsolete

This command performs two functions. First it deletes the object from the recovery catalog. Second, using the ADSMConnect Agent, it starts a session with the ADSM server and *deactivates* the object on the server. Figure 101 illustrates the change to the recovery catalog. The output from the **report obsolete** command no longer shows the **DFM8_40_1/Level_0** backup piece. It, along with the backup set it belonged, to have been deleted.

```
RMAN> report obsolete ;

RMAN-03022: compiling command: report
RMAN-06280: Report of obsolete backup sets and datafile copies
RMAN-06281: Type                 Recid  Stamp     Filename
RMAN-06282: -------------------- ------ --------- -----------------------
RMAN-06284: Backup Set           33     328200301
RMAN-06285: Backup Piece         33     328200298 DFM8_42_1/Level_2
RMAN>
```

Figure 101. RMAN Report Obsolete Command Output after Deletion

The change on the ADSM server can be confirmed by issuing the administrative **select** command again. Figure 102 show that the **DFM8_40_1/Level_0** backup object is now an inactive backup version.

```
adsm> select state, ll_name, backup_date from backups where node_name='SEVERN'
   ⋮

             STATE     LL_NAME                    BACKUP_DATE
------------------     ------------------     ------------------
    ACTIVE_VERSION     DFM8_39_1/Level_0             1998-03-17
                                                14:36:18.000000
    ACTIVE_VERSION     DFM8_41_1/Level_2             1998-03-17
                                                15:01:45.000000
    ACTIVE_VERSION     DFM8_42_1/Level_2             1998-03-17
                                                15:03:11.000000
    ACTIVE_VERSION     DFM8_43_1/Level_2             1998-03-17
                                                15:15:37.000000
    ACTIVE_VERSION     DFM8_44_1/Level_2             1998-03-17
                                                15:17:06.000000
  INACTIVE_VERSION     DFM8_40_1/Level_0             1998-03-17
                                                14:37:50.000000
```

Figure 102. Displaying Inactive RMAN Objects on ADSM

This inactive backup version will be expired from the ADSM server the next time that an *expire inventory* process is run.

Management Class

The above example of expiring a backup on the ADSM server is based on using a management class that does not retain inactive backup versions, as discussed in 8.4.2.4, “Customize the Management Class” on page 172. Use of a management class for ADSMConnect Agent backup objects that is configured to retain inactive backup versions will prevent RMAN backup objects from being expired from the ADSM server.

8.7.5 Scheduled Backups

This example illustrates a scheduled backup using the ADSM scheduled client. The example is an incremental level 2 backup performed daily on weekdays. Similar scheduled backups could be created to perform whole or incremental level 0 backups at weekly or monthly intervals as appropriate.

The example is based on a script that executes a number of RMAN commands and writes output to a log file with a date and time stamp. The ADSM scheduled client runs with root authority. Therefore the script executed by the scheduler must be in a filesystem that is accessible to root with appropriate execution permissions. This example involved creating:

- A script executed by the scheduled client with root authority
- A script that executed the RMAN commands
- A schedule on the ADSM server to execute the script

8.7.5.1 Scheduler Script

Figure 103 illustrates a shell script named **/tmp/rman_inc2** that is executed by the ADSM scheduled client.

```
#!/usr/bin/ksh
export  backlog=$(/usr/bin/date +%m%d%y)
export  backdate=$(/usr/bin/date +%m"/"%d"/"%y)
export  backstart=$(/usr/bin/date +%H":"%M":"%S)

echo "                                                              " >> /tmp/rman_inc2.$backlog
echo "#### Start of RMAN INC_2 Backup ($backdate $backstart) ####" >> /tmp/rman_inc2.$backlog
echo "                                                              " >> /tmp/rman_inc2.$backlog

su - oracle -c rmancmd "'rman/script/inc2'"  >> /tmp/rman_inc2.$backlog


export  backstop=$(/usr/bin/date +%H":"%M":"%S)
echo "                                                            " >> /tmp/rman_inc2.$backlog
echo "#### End of RMAN INC_2 Backup ($backdate $backstop) ####" >> /tmp/rman_inc2.$backlog
echo "                                                            " >> /tmp/rman_inc2.$backlog
```

Figure 103. rman_inc2 Scheduled Script

This script sets up an environment for creating a date and time stamped log file and executes a second script containing the RMAN commands. A number of environment variables are set based on the date, backup start, and stop times. These are echoed in a log file with a name of rman_inc2.*MMDDYY*, where MMDDYY is the current date.

Having created the log file, the script then switches to the Oracle user account, using the AIX **su** command, and executes an RMAN script named inc2 in the /home/oracle/rman/script directory. The script is executed by using the **rmancmd** alias discussed in 8.6, “Backup and Recovery Examples” on page 187.

8.7.5.2 RMAN Script

Figure 104 illustrates the RMAN script executed by the */tmp/rman_inc2* script.

```
# script name:   inc2
#
# tasks       this script executes level 2 incremental backup
#              for whole database
#
resync catalog;                              1
run {
execute script alloc_adsm;                   2
backup                                       3
     incremental level 2
     format '%d_%s_%p/Level_2_DB'
     tag Level_2_DB
     (database) ;
backup                                       4
     format '%d_%s_%p/Level_2_CF'
     tag Level_2_CF
     (current controlfile);
}
```

Figure 104. inc2 RMAN Script

This script executes four RMAN commands:

1 The first command resynchronizes the recovery catalog with the target database as a precaution in case any changes have been made to the database schema. This command must be executed before the run statement.

2 The second command allocates a channel to ADSM.

3 The third command performs the level 2 backup of the database. It uses a format of *%d_%s_%p/Level_2_DB*. This will be the name of the object stored on ADSM.

4 The fourth command performs a separate backup of the current control file following the database backup. This uses a format of *%d_%s_%p/Level_2_CF*. Again this will be the name of the object stored on ADSM. The separate control file backup is performed so that it is stored on ADSM with a unique, easily identified object name.

8.7.5.3 Define ADSM Schedule

The final configuration step is to define a schedule on ADSM. A command schedule is defined to execute the RMAN script created in 8.7.5.1, “Scheduler Script” on page 216. Figure 105 on page 218 illustrates a query of the schedule after it was defined.

```
adsm> query schedule standard rman_inc2 format=detailed

            Policy Domain Name: STANDARD
                 Schedule Name: RMAN_INC2
                   Description: RMAN_INC_Level_2_DB_Backup
                        Action: Command
                       Options:
                       Objects: /tmp/rman_inc2
                      Priority: 1
                  Start Date/Time: 03/19/1998 02:00:00
                      Duration: 4 Minute(s)
                        Period: 1 Day(s)
                   Day of Week: Weekday
                    Expiration:
Last Update by (administrator): TIM
         Last Update Date/Time: 03/18/1998 10:45:40
```

Figure 105. ADSM RMAN Command Schedule

The Action field specifies that an AIX command will be executed and the Objects field specifies the command, the rman_inc2 shell script. It is a daily schedule that executes at 2:00 a.m. on weekdays.

8.7.5.4 Checking Schedule Results

Having established the schedule, the /tmp/rman_inc2 script will be executed daily. This script creates date stamped logs in the /tmp filesystem:

```
/tmp % ls -l rman_inc2.*
-rw-r--r--   1 root     system      2873 Mar 18 02:25 rman_inc2.031898
-rw-r--r--   1 root     system      2873 Mar 19 02:32 rman_inc2.031998
-rw-r--r--   1 root     system      2873 Mar 20 02:27 rman_inc2.032098
  ⋮
```

Figure 106 illustrates extracts from one of the logs with details of the RMAN jobs performed.

```
#### Start of RMAN INC_2 Backup (03/18/98 02:15:01) ####

Recovery Manager: Release 8.0.3.0.0 - Production

RMAN-06005: connected to target database: DFM8
RMAN-06008: connected to recovery catalog database
RMAN> # script name:   inc2
2> #
3> # tasks  this script executes level 2 incremental backup
4> #         for whole database
5> #
6> resync catalog;

7> run {
8> execute script alloc_adsm;
9> backup
10>  incremental level 2
11>  format '%d_%s_%p/Level_2_DB'
12>  tag Level_2_DB
13>  (database);
14> backup
15>  format '%d_%s_%p/Level_2_CF'
16>  tag Level_2_CF
17>  (current controlfile);
18> }
19>
  ⋮
```

Figure 106 (Part 1 of 2). Sample RMAN Scheduled Backup Log

```
RMAN-03022: compiling command: resync
RMAN-03023: executing command: resync
RMAN-08002: starting full resync of recovery catalog
RMAN-08004: full resync complete

RMAN-03021: executing script: alloc_tape
   ⋮
RMAN-08030: allocated channel: adsm
RMAN-08500: channel adsm: sid=14 devtype=SBT_TAPE

RMAN-03022: compiling command: backup
RMAN-03023: executing command: backup
RMAN-08008: channel adsm: starting datafile backupset
RMAN-08502: set_count=52 set_stamp=328356262
RMAN-08010: channel adsm: including datafile 1 in backupset
   ⋮
RMAN-08503: piece handle=DFM8_52_1/Level_2_DB comment=API Version 0.1,MMS Version 2.1.6.0
RMAN-03023: executing command: partial resync
RMAN-08003: starting partial resync of recovery catalog
RMAN-08005: partial resync complete

RMAN-03022: compiling command: backup
   ⋮
RMAN-08011: channel adsm: including current controlfile in backupset
RMAN-08013: channel adsm: piece 1 created
RMAN-08503: piece handle=DFM8_53_1/Level_2_CF comment=API Version 0.1,MMS Version 2.1.6.0
   ⋮

Recovery Manager complete.

#### End of RMAN INC_2 Backup (02/18/98 02:24:08) ####
```

Figure 106 (Part 2 of 2). Sample RMAN Scheduled Backup Log

The log contains a banner with the start date and time, details of the RMAN functions run, and an ending banner with the completion date and time.

8.8 Recovery Catalog Availability

The recovery catalog is essential to the operation of RMAN. If a target database must be recovered, RMAN queries the recovery catalog to determine which objects to restore and recover. Without a recovery catalog, it is difficult to know which objects to restore from the ADSM server. This section covers the following recovery catalog topics:

- Recovery catalog resynchronization
- Exporting the recovery catalog
- Recovery catalog archiving with ADSM
- Recovery catalog backup with RMAN

8.8.1 Recovery Catalog Resynchronization

If the structure of a target database is changed, such as newly created tablespaces or data files, a log switch, or log archive operation, the recovery catalog is not updated automatically. It is recommended that the **resync catalog** RMAN command be run on a regular basis to resynchronize the target database with the recovery catalog. However, most RMAN commands such as backup, copy, restore, or switch update the recovery catalog automatically.

The ADSM scheduling functions can be used to include the RMAN resync catalog command within scheduled backup operations. 8.7.5, “Scheduled Backups” on page 216 describes a scheduled backup containing a catalog resynchronization.

8.8.2 Exporting the Recovery Catalog

One option for protecting the recovery catalog is to periodically export the catalog database instance to a file and then archive that file to ADSM.

The Oracle **export** command is used to export a database. The following example illustrates a full export of the recovery catalog database to a file called ***rcvcat.exp*** followed by an ADSM archive operation of the resulting export file:

```
% exp userid=system/manager full=y consistent=y file=/tmp/rcvcat.exp
  ⋮
% dsmc archive /tmp/rcvcat.exp -desc="RMAN recovery catalog export"
```

To restore the recovery catalog database, the export is retrieved from the ADSM server and imported to the recovery catalog database:

```
% dsmc retrieve /tmp/rcvcat.exp
  ⋮
% imp userid=system/manager full=y file=/tmp/rcvcat.exp
```

This example uses the ADSM archive function to archive the database export. The backup function could also be used.

8.8.3 Database Archive with ADSM

An alternative to exporting the recovery catalog is to use ADSM to perform an offline archive of the catalog. This technique can be used only if the database is installed on filesystems. Raw logical volumes are not supported. To perform an offline archive of the recovery catalog, it must first be shut down:

```
SVRMGR> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SVRMGR>
```

The recovery catalog database used in these examples was installed by using the Oracle Flexible Architecture (OFA), which creates the database in filesystems /u01, /u02, and /u03. After the recovery catalog database is shut down, the related recovery catalog files are archived by using the ADSM dsmc archive command from an AIX command prompt:

```
% dsmc archive -su=yes /u01/ /u02/ /u03/ -desc="RMAN Catalog Backup"
```

To restore the recovery catalog database files, you shut down the recovery catalog database if it is open. Then retrieve the files from the ADSM server. From the AIX command line enter:

```
% dsmc retrieve -su=yes -rep=all /u01/ /u02/ /u03/ -desc="RMAN Catalog Backup"
```

After the recovery catalog database files have been restored, you can start up the recovery catalog database. An Oracle recovery operation is not necessary because the database was closed during the archive operation.

8.8.4 Recovery Catalog Backup with RMAN

Because the recovery catalog is stored in an ordinary Oracle8 database instance, RMAN could be used to back up the recovery catalog.

One approach, a peer-to-peer method, requires two Oracle8 databases on separate machines, with RMAN on each database instance using the other database for its recovery catalog. This peer-to-peer configuration would work for two database instances installed on separate systems. However, it is not a practical method if both Oracle8 database instances are on a single system or if there are more than two database instances. For multiple instances it is desirable to have a single recovery catalog to ease the task of administration.

If a single recovery catalog is used for all target databases, backing up the recovery catalog with RMAN presents some challenges. The recovery catalog can be backed up with RMAN, but the target database (the recovery catalog instance) and the recovery catalog would be the same database instance. One solution to this problem is to use RMAN with the **nocatalog** option to back up the recovery catalog. Using this option, information about backups will be stored in the target database control file, which in this case is the recovery catalog database control file.

Follow these steps to back up and recover the CAT8 database:

1. Back up recovery catalog database
2. Back up control file
3. Delete database
4. Recover database control
5. Restore database using current control file
6. Recover database

In the example we connect to the target database CAT8 (the recovery catalog) in **nocatalog** mode:

```
% rman target system/manager nocatalog

Recovery Manager: Release 8.0.3.0.0 - Production

RMAN-06009: using target database controlfile instead of recovery catalog
RMAN-06005: connected to target database: CAT8
RMAN>
```

8.8.4.1 Back up Recovery Catalog Database

The first step is to perform a full offline backup of the recovery catalog database. The database must first be closed:

```
SVRMGR> alter database close;
Statement processed.
SVRMGR>
```

Once the database is closed, it can be backed up. Figure 107 on page 222 illustrates the log from the database backup containing the commands executed.

```
Recovery Manager: Release 8.0.3.0.0 - Production

RMAN-06009: using target database controlfile instead of recovery catalog
RMAN-06005: connected to target database: CAT8
RMAN> run {

2> allocate channel adsm1 type 'SBT_TAPE' \
parms 'ENV=(DSMI_DIR=/usr/lpp/adsmagent/aob,DSMO_FS=adsmorc,\
DSMI_LOG=/var/adsm/log,DSMO_PSWDPATH=/usr/lpp/adsmagent/aob)';
3> backup format 'whole_%d%s' (database) ;
4> }

RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: adsm1
RMAN-08500: channel adsm1: sid=11 devtype=SBT_TAPE

RMAN-03022: compiling command: backup
RMAN-03023: executing command: backup
RMAN-08008: channel adsm1: starting datafile backupset
RMAN-08502: set_count=3 set_stamp=326278413
RMAN-08010: channel adsm1: including datafile 1 in backupset
   ⋮
RMAN-08013: channel adsm1: piece 1 created
RMAN-08503: piece handle=whole_CAT84 comment=API Version 0.1,MMS Version 2.1.6.0
RMAN-08031: released channel: adsm1

Recovery Manager complete.
```

Figure 107. RMAN Recovery Catalog Backup Using nocatlog Option

8.8.4.2 Back up Control File

The details of the backup operation are written to the database control file after the successful completion of the backup. These details are required to restore the backup. The next step is to copy the control file and then back it up to ADSM. Without the recovery catalog, a copy of the current control file is essential. Figure 108 illustrates the RMAN job that copies the current control file to a temporary file, /tmp/thectrl.

```
Recovery Manager: Release 8.0.3.0.0 - Production

RMAN-06009: using target database controlfile instead of recovery catalog
RMAN-06005: connected to target database: CAT8
RMAN> run {

2> allocate channel disk1 type disk;
3> copy  current controlfile to '/tmp/thectrl' ;
4> }

RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: disk1
RMAN-08500: channel disk1: sid=10 devtype=DISK

RMAN-03022: compiling command: copy
RMAN-03023: executing command: copy
RMAN-08027: channel disk1: copied current controlfile
RMAN-08505: output filename=/tmp/thectrl
RMAN-08031: released channel: disk1
```

Figure 108 (Part 1 of 2). RMAN Copy Control File Job

```
Recovery Manager complete.
```

Figure 108 (Part 2 of 2). RMAN Copy Control File Job

Next back up the control file copy to ADSM to ensure that the RMAN backup and a copy of the control file with details of the backup are backed up together on ADSM. Figure 109 illustrates the control file copy being backed up to ADSM.

```
% dsmc sel /tmp/thectrl
ADSTAR Distributed Storage Manager
Command Line Backup Client Interface - Version 3, Release 1, Level 0.1
(C) Copyright IBM Corporation, 1990, 1997, All Rights Reserved.

Selective Backup function invoked.

Node Name: BLACK
Please enter password for node "BLACK":

Session established with server DB2: AIX-RS/6000
  Server Version 3, Release 1, Level 0.1
  Server date/time: 02/26/1998 09:13:38  Last access: 02/26/1998 09:13:10

Normal File-->           421,888 /tmp/thectrl  Sent
Selective Backup processing of '/tmp/thectrl' finished without failure.
Total number of objects inspected:        1
Total number of objects backed up:        1
Total number of objects updated:          0
Total number of objects rebound:          0
Total number of objects deleted:          0
Total number of objects failed:           0
Total number of bytes transferred:   412.02 KB
Data transfer time:                    0.63 sec
Network data transfer rate:          645.29 KB/sec
Aggregate data transfer rate:         83.81 KB/sec
Objects compressed by:                    0%
Elapsed processing time:           00:00:04
oracle@black %
```

Figure 109. ADSM Backup of RMAN Control File Copy

8.8.4.3 Delete Database

With a secure backup of the database and control file on ADSM, the recovery catalog can be removed. First shut down the recovery catalog database:

```
SVRMGR> shutdown immediate
ORA-01109: database not open
Database dismounted.
ORACLE instance shut down.
SVRMGR>
```

Then delete the data files, control files, and the control file copy to render the database inoperable:

```
% rm /u0?/oradata/CAT8/*
% rm /tmp/thectl
```

8.8.4.4 Recover Control File

The control file must be restored before the database can be restored. Two steps involved are: restoring the control file copy from ADSM and replicating the control file by using RMAN. Figure 110 illustrates the control file copy being restored from ADSM.

```
% dsmc restore /tmp/thectrl
ADSTAR Distributed Storage Manager
Command Line Backup Client Interface - Version 3, Release 1, Level 0.1
(C) Copyright IBM Corporation, 1990, 1997, All Rights Reserved.

Restore function invoked.

Node Name: BLACK
Please enter password for node "BLACK":

Session established with server DB2: AIX-RS/6000
  Server Version 3, Release 1, Level 0.1
  Server date/time: 02/26/1998 09:18:16  Last access: 02/26/1998 09:13:38

Restoring         421,888 /tmp/thectrl  Done

Restore processing finished.

Total number of objects restored:         1
Total number of objects failed:           0
Total number of bytes transferred:   412.09 KB
Data transfer time:                    0.35 sec
Network data transfer rate:        1,148.69 KB/sec
Aggregate data transfer rate:         83.70 KB/sec
Elapsed processing time:           00:00:04
oracle@black %
```

Figure 110. ADSM Restore of RMAN Control File Copy

Once the control file copy is restored, the database must be started in nomount mode so that no attempt is made to open any of the missing files:

```
SVRMGR> startup nomount ;
ORACLE instance started.
Total System Global Area       5536980 bytes
Fixed Size                       42592 bytes
Variable Size                  4150900 bytes
Database Buffers                819200 bytes
Redo Buffers                    524288 bytes
SVRMGR>
```

Next RMAN is used to replicate the current control file from the restored copy of the control file. The step restores the three control files from the /tmp/thectrl copy control file. Figure 111 illustrates the replicate job.

```
Recovery Manager: Release 8.0.3.0.0 - Production

RMAN-06009: using target database controlfile instead of recovery catalog
RMAN-06006: connected to target database: CAT8 (not mounted)
RMAN> run {
2> allocate channel disk1 type disk;
3> replicate controlfile  from '/tmp/thectrl';
4> }
```

Figure 111 (Part 1 of 2). RMAN Replicate Control File Job

```
RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: disk1
RMAN-08500: channel disk1: sid=11 devtype=DISK

RMAN-03022: compiling command: replicate
RMAN-03023: executing command: replicate
RMAN-08058: replicating controlfile
RMAN-08506: input filename=/tmp/thectrl
RMAN-08505: output filename=/u01/oradata/CAT8/control01.ctl
RMAN-08505: output filename=/u02/oradata/CAT8/control02.ctl
RMAN-08505: output filename=/u03/oradata/CAT8/control03.ctl
RMAN-08031: released channel: disk1

Recovery Manager complete.
```

Figure 111 (Part 2 of 2). RMAN Replicate Control File Job

The current control file for the database instance now contains details of the most recent backup. This can be confirmed by starting RMAN in nocatalog mode and using the **list** command to list the backup information in the newly replicated control file. Before this can be done, the database must be mounted:

```
SVRMGR> alter database mount;
Statement processed.
SVRMGR>
```

Figure 112 illustrates the RMAN list command used to list the available backup sets from the current control file.

```
Recovery Manager: Release 8.0.3.0.0 - Production

RMAN-06009: using target database controlfile instead of recovery catalog
RMAN-06006: connected to target database: CAT8 (not mounted)
RMAN> list backupset of database;
RMAN-03022: compiling command: list
RMAN-06230: List of Datafile Backups
RMAN-06231: Key     File Type         LV Completion_time Ckp SCN    Ckp Time
RMAN-06232: ------- ---- ------------ -- --------------- ---------- --------
RMAN-06233: 1       1    Full            25-FEB-98       63293      25-FEB-98
RMAN-06233: 3       1    Full            26-FEB-98       63305      26-FEB-98
RMAN-06233: 1       2    Full            25-FEB-98       63293      25-FEB-98
RMAN-06233: 3       2    Full            26-FEB-98       63305      26-FEB-98
   ⋮
RMAN-06233: 1       9    Full            25-FEB-98       63293      25-FEB-98
RMAN-06233: 3       9    Full            26-FEB-98       63305      26-FEB-98
RMAN>
```

Figure 112. List Backupsets from Current Control File

8.8.4.5 Restore Database Using Current Control File

Next the database is restored from ADSM. Starting RMAN in nocatalog mode and executing a **restore** command will restore the database. The required information is obtained from the current control file (Figure 113 on page 226).

```
Recovery Manager: Release 8.0.3.0.0 - Production

RMAN-06009: using target database controlfile instead of recovery catalog
RMAN-06006: connected to target database: CAT8 (not mounted)
RMAN> run {
2> allocate channel adsm1 type 'SBT_TAPE'
parms 'ENV=(DSMI_DIR=/usr/lpp/adsmagent/aob,DSMO_FS=adsmorc,DSMO_PSWDPATH=/usr/lpp/adsmage
nt/aob)';
3> restore (database);
4> }

RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: adsm1
RMAN-08500: channel adsm1: sid=9 devtype=SBT_TAPE

RMAN-03022: compiling command: restore
RMAN-03023: executing command: restore
RMAN-08016: channel adsm1: starting datafile backupset restore
RMAN-08502: set_count=3 set_stamp=326278413
RMAN-08019: channel adsm1: restoring datafile 1
   ⋮
RMAN-08511: piece handle=whole_CAT83 params=NULL
RMAN-08024: channel adsm1: restore complete
RMAN-08031: released channel: adsm1

Recovery Manager complete.
```

Figure 113. RMAN Recovery Catalog Restore with nocatlog Option

8.8.4.6 Recover Database

After the database is restored it must be recovered. To check the status use the following server manager command:

```
SVRMGR>select LOG_MODE, CONTROLFILE_TYPE, OPEN_RESETLOGS from v$database;

LOG_MODE     CONTROL OPEN_RESETL
------------ ------- -----------
NOARCHIVELOG BACKUP  ALLOWED
1 row selected.
```

Note that the backup control file is being used. The database must be recovered by using this backup control file:

```
SVRMGR>recover database until cancel using backup controlfile ;
Specify log: {<RET>=suggested | filename | AUTO | CANCEL }
cancel
Media recovery cancelled.
```

The recover command prompts for the redo logs to be used for recovery. As we are restoring a consistent full backup, *cancel* can be entered because there are no logs to apply. This completes the recovery; the database is now using the current control file.

The database can now be opened by using the resetlogs option:

```
SVRMGR> alter database open resetlogs ;
Statement processed.
SVRMGR>
SVRMGR> select LOG_MODE,CONTROLFILE_TYPE,OPEN_RESETLOGS from v$database;
LOG_MODE     CONTROL OPEN_RESETL
------------ ------- -----------
NOARCHIVELOG CURRENT NOT ALLOWED
SVRMGR>
```

The recovery catalog is now available for use. At this point it is important that a full offline backup is run immediately as the resetlogs option invalidates earlier backups.

Chapter 9. SQL-BackTrack

This chapter describes the SQL-BackTrack product from BMC Software Inc. and its use with ADSM. SQL-BackTrack is designed to provide a complete backup solution for database backups. It offers facilities such as true incremental backup, object level backup and recovery, table and column level recovery, data compression, and encryption. SQL-BackTrack provides additional modules that enable integration directly with ADSM. SQL-BackTrack supports Oracle, Sybase, and Informix databases.

SQL-BackTrack for Oracle supports the following platforms:

- AIX 3.2 and above
- SunOS 4.1.x and Solaris 2.2 and above
- HP-UX 9.0 and above
- Digital UNIX 3.2 and 4.0
- SINIX 5.42
- Sequent PTX 4.2.1
- Windows NT 3.51 and 4.0

SQL-BackTrack for Sybase supports the following platforms:

- AIX 3.2 and 4.1 to 4.3
- HP-UX 9.04, 10.x, and 11.0
- Digital UNIX 3.2 and 4.0+
- SunOS 4.1.3 or Solaris 2.2 to 2.6
- NCR UNIX 2.0 and 3.0

SQL-BackTrack for Informix supports the following platforms:

- AIX 4.1 to 4.3
- HP-UX 9.04, 10.x, and 11.0
- SunOS 4.1.3 or Solaris 2.4 to 2.6
- SINIX 5.42

Note

A wide range of database levels are supported on the various platforms. Please contact BMC Software Inc. for a current matrix of supported platforms and database levels.

This chapter focuses on SQL-BackTrack for Oracle and covers the following topics:

1. SQL-BackTrack for Oracle overview
2. SQL-BackTrack features
3. SQL-BackTrack installation and configuration
4. ADSM OBSI module installation
5. SQL-BackTrack catalog
6. Backup and recovery examples

For this project a beta copy of SQL-BackTrack Version 3.0.1 for Oracle 7.3 and 8 was used on AIX 4.1.4 with Oracle 7.3.4 and Oracle 8.0.3 databases.

9.1 SQL-BackTrack for Oracle Overview

SQL-BackTrack provides backup and recovery for Oracle7 and Oracle8 databases. This section presents a brief overview of the architecture and the components involved in backing up Oracle databases with SQL-BackTrack and ADSM.

9.1.1 Architecture

SQL-BackTrack provides a complete backup solution for Oracle database backups. It can be used to perform database exports; online or offline full and incremental backups; database, tablespace, and individual data file recovery; and logical table level recovery. SQL-BackTrack can store database backup objects on a variety of devices: local disk, locally attached tape drives, and external storage management products such as ADSM.

Communication between SQL-BackTrack and the physical device is enabled through *Open Backup Stream Interface (OBSI) modules.* SQL-BackTrack can manage backup and recovery directly to disk or locally attached tape, using the SQL-BackTrack disk and tape OBSI modules provided with the base product. To exploit external storage management products such as ADSM, additional OBSI modules must be installed. The *ADSM OBSI module* provides the connection between SQL-BackTrack running on an Oracle instance and a network-attached ADSM server (Figure 114).

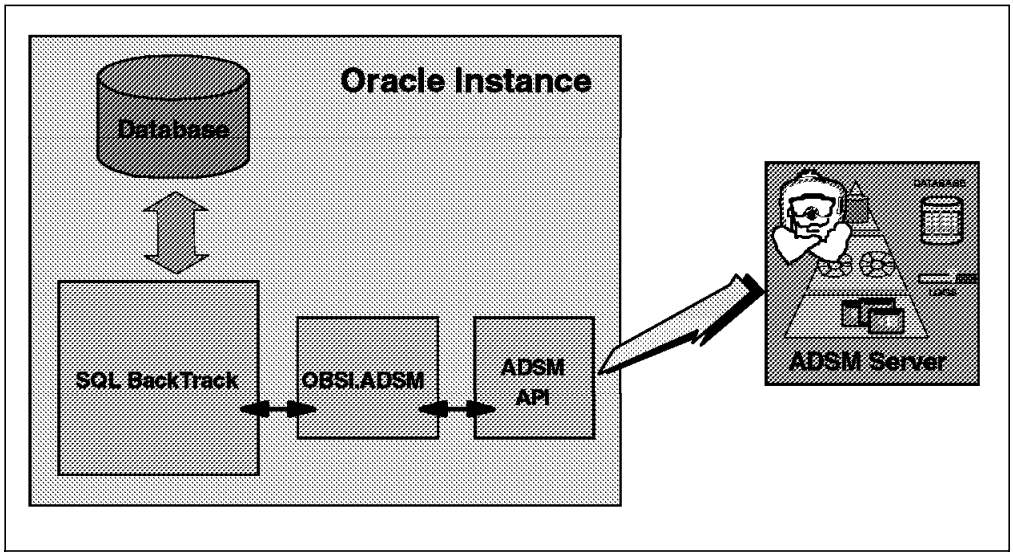


Figure 114. SQL-BackTrack and ADSM Architecture

SQL-BackTrack performs the backup and recovery functions on the Oracle database instance. The backup objects are written to, or read from, the ADSM OBSI module (OBSI.ADSM). The ADSM OBSI module uses the ADSM API to send or receive the data from the ADSM server. Multiple ADSM servers can be utilized to improve overall backup performance.

9.1.2 System Components

SQL-BackTrack consists of a number of components that interact during the backup and restore processes:

- obacktrack program

  The **obacktrack** program is the menu-driven interface to SQL-BackTrack. It is used to create and execute backups and exports and to recover or import data. When executing, **obacktrack** prompts for all relevant options and saves the information in the SQL-BackTrack catalog directory.

- SQL-BackTrack catalog

  The SQL-BackTrack catalog is a collection of files and directories that hold all information regarding profiles and backup pools (Figure 115).

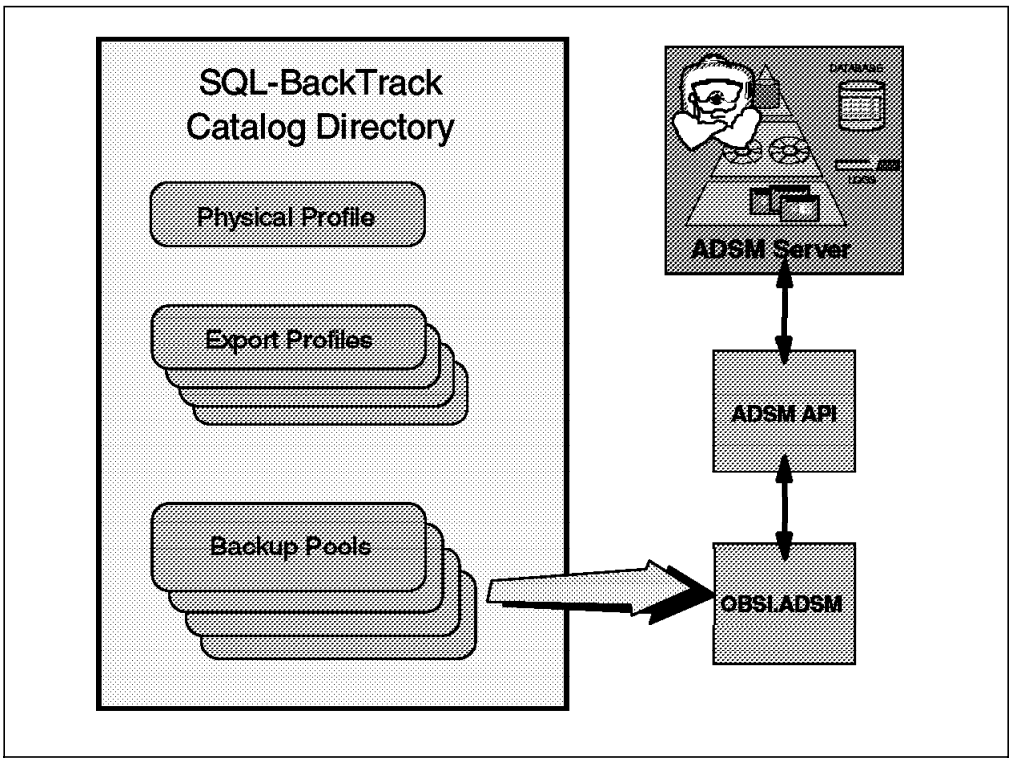


Figure 115. SQL-BackTrack Catalog Contents

  The catalog contains only one physical profile per database but can contain multiple export profiles and SQL-BackTrack backup pools. Unlike EBU and RMAN, there is no need for a dedicated Oracle database instance to maintain the SQL-BackTrack backup catalog. Every host that has SQL-BackTrack installed has its own catalog. The catalog contains information for all Oracle database instances being managed by SQL-BackTrack on that host. The **obacktrack** program automatically synchronizes the catalog with the database before every backup operation.

- Backup pools

  An SQL-BackTrack backup pool describes a backup device and its physical characteristics. The backup device can be disk, a locally attached tape, or ADSM through the ADSM OBSI module. Before a backup pool is configured, the physical device on which the backup objects will be stored must be defined.

ADSM Storage Pools

The SQL-BackTrack backup pool is not the ADSM storage pool. The SQL-BackTrack backup pool points to ADSM as the backup device.

- Profiles

  In order to use SQL-BackTrack you have to create physical and export profiles. A profile describes the characteristics of the operation to be performed, including the destination SQL-BackTrack backup pool. Profiles are stored in the SQL-BackTrack catalog.

9.2 Features

This section introduces the main functions that can be performed with SQL-BackTrack: physical backups, archive log backups, database exports, database recovery, database imports, logical table extraction, and additional administrator functions.

9.2.1 Physical Backups

A physical backup is a backup of the following database files:

- Data files
- Redo logs
- Archived redo logs
- Control files
- Initialization parameter files (init.ora)

With SQL-BackTrack there are different kinds of physical backup:

- Cold backup

  During a cold backup, SQL-BackTrack shuts down the database and backs up the database files. After the backup is completed, SQL-BackTrack starts up the database. A cold backup can be either full or incremental.

- Online backup

  An online backup, or hot backup, is performed while the database is open and being used. The database must be running in ARCHIVELOG mode to perform online backups. An online backup can be either full or incremental.

- Selective backup

  With a selective, or partial backup, individual tablespaces or data files can be backed up during cold and online backups. Selective backups can only be performed if the database is running in ARCHIVELOG mode. During online backups, tablespaces may be either online or offline. If the tablespace is online, you can access it during the backup. If the tablespace is offline, you cannot access it.

- Incremental backup

  With physical incremental backups, SQL-BackTrack backs up only the data blocks that have changed since the last physical backup. SQL-BackTrack supports incremental backups at levels 0 through 9. Level 0 is a full backup, level 9 is an incremental backup, and levels 1 through 8 are intermediate incremental backups. Levels 1 through 8 back up all the changes since the last backup performed at a lower level. SQL-BackTrack automatically restores and applies incremental backups when the database or selected

database object is recovered. SQL-BackTrack provides incremental backups for both Oracle7 and Oracle8 databases.

Physical backup provides the fastest way to recover from media failure. SQL-BackTrack performs fast physical backup by providing compression and incremental backups.

An Oracle database can be run in either ARCHIVELOG or NOARCHIVELOG mode. It is recommended that the database be run in ARCHIVELOG mode, which provides the following capabilities when using SQL-BackTrack:

- Online and offline physical backups can be performed.
- Point-in time recovery can be performed. Recovery can be performed to a specific date and time or alternatively to an Oracle System Change Number (SCN).
- Automatic backup of the archived redo logs is performed during physical backup. Once the logs are backed up, SQL-BackTrack can rename or delete the redo logs.
- The required archived redo logs are automatically restored and then applied during recovery.
- The chances of an error occurring during backup and recovery are reduced. SQL-BackTrack for Oracle issues all necessary Oracle server commands.

If the database is run in NOARCHIVELOG mode, online and incremental backups cannot be performed. In addition recovery can be performed only to the last backup. Any database transactions after the backup are lost.

9.2.2 Archive Log Backups

SQL-BackTrack manages the archived redo log backups and provides additional features for archive log management. SQL-BackTrack automatically backs up the archive logs when you perform a physical backup. It also tracks the Oracle SCN range of each archivelog backup to enable recovery to a specific SCN. SQL-BackTrack monitors archived logs and backs up these logs before the archive log destination fills up. You can choose to delete or rename the original archive logs after the backup operation.

9.2.3 Database Exports

The SQL-BackTrack export operation invokes the Oracle **exp** utility. An export backup is also termed a **logical backup** because it can create backup objects at the SQL level. Files created by export backup can be read only by the import utility. SQL-BackTrack export supports full, incremental, user, and tablespace exports. SQL-BackTrack can write exports directly to an ADSM server without the need for an intermediate file.

Unlike physical backup, export does not provide point-in-time recovery. You can not import a table and roll it forward with the redo logs. An export is usually slower than a physical backup.

9.2.4 Database Recovery

SQL-BackTrack provides the capability to perform the following types of recovery: physical recovery, point-in-time recovery, redo log recovery, and control file recovery. A guided recovery feature assists in determining what is wrong with the database and which restore and recovery operation needs to be performed.

9.2.4.1 Physical Recovery

During a physical recovery SQL-BackTrack uses physical backup data to restore and recover the database or object being recovered. The **obacktrack** program guides you through the recovery process. During the guided recovery it performs the following actions:

- Detects any missing database files: data files, online redo log files, control files, or parameters files
- Presents the recovery options appropriate for the situation and lets you choose the type of recovery
- Issues the recovery commands
- Restores the required database files
- Restores and recovers archived log files
- Recovers control files and parameter files as necessary
- Brings the database online

9.2.4.2 Point-in-Time Recovery

You can perform point-in-time recovery if the database is operating in ARCHIVELOG mode. A point-in-time recovery recovers the database to a specified date and time.

A **change base** recovery can also be performed. This recovery uses the Oracle SCN instead of a date and time to perform the recovery. You can perform point-in-time or SCN recovery only at the database level.

SQL-BackTrack simplifies the recovery process by:

- Tracking the time and date of each backup
- Tracking all archive logs by time and system change number
- Applying the archive logs to reach the specific point in time

During point-in-time recovery, the data files can be restored to their original location or to an alternative location.

9.2.4.3 Archive Log and Online Redo Log Recovery

When the database or an object is recovered, SQL-BackTrack automatically recovers and applies archive logs and online redo logs.

If the archive log file system fills up during the recovery, SQL-BackTrack automatically applies the restored archive logs and deletes them. Thus large numbers of archive logs can be recovered without the need to allocate additional disk space to perform this task.

9.2.4.4 Control File and Parameter File Recovery

SQL-BackTrack recovers the control files and parameter files (init.ora) automatically during the recovery operation. No additional actions are required to recover them.

9.2.5 Database Imports

You can import data from an export you have taken with SQL-BackTrack. You cannot import data from an export that was not taken by SQL-BackTrack. Use of the import enables you to recover a single table from a full database export. The import is done directly from ADSM without creating an intermediate file.

9.2.6 Logical Table Extraction

SQL-BackTrack Version 3 introduces a new function that enables individual tables or columns to be recovered. This logical table extraction feature recovers a specific table or set of tables from a physical backup and applies the archive logs to recover the tables to a specific point in time. With logical extraction you can recover:

- One or more tables for each tablespace
- A specific column of a table
- A point-in-time table
- A table into another database

Logical extraction can write the extracted data directly back into the database, into an SQL file that can be subsequently applied to the database, or into a plain text file.

Note

Inserting the data directly into the database does not overwrite the data if it still exists in the database. A situation such as this results in duplicate data in the database.

You cannot do logical extraction with export files. Logical extraction must be enabled during the backup profile configuration.

9.2.7 Administrator Functions

In addition to the main backup and recovery functions there a number of additional administrative functions. These include backup expiration, dry-run operations, archive log monitoring, report generation, and script creation.

9.2.7.1 Backup Expiration

During backup configuration you can decide how long to retain the backups and define the criteria for expiring backups. SQL-BackTrack expires backups automatically based on the defined criteria. You can also expire backups manually.

The expiration parameters are based on the number of days you want to keep your backup and the number of complete copies to keep. For expiring an object, both conditions must be met.

9.2.7.2 Dry-run Operations

When a backup, recover, export, or import is performed, you can perform a dry run before the real operation.

Backup and export dry runs connect to the target database and read through the database data. A dry-run backup checks that the user has the privileges to perform the backup and that the backup profile is valid. A dry run does not:

- Affect the state of the database
- Verify that the backup device is available for write operations
- Write any data to the backup media

A recovery dry run tests recovery without affecting the state of the database or writing to the database. You can perform a dry run of physical recovery, logical extraction, or logical recovery. The dry-run recover checks that the user has the privileges to perform the recover, the backup catalog is valid, and the media is available. However, it does not load and read the physical tape volumes.

9.2.7.3 Archive Log Monitoring

SQL-BackTrack automatically backs up the archive logs during every physical backup. For databases that generate a high volume of archive logs, you may need to back up the archive log more frequently than the regular database backups. SQL-BackTrack supports archive log monitoring to automate this process.

The **dtoarchmon** process monitors the archive log destination and performs backups when specific thresholds are met. Thresholds can be the number of archive logs, the space utilization of the archive log destination, or a combination of both. These archive log options are specified during the physical profile configuration.

9.2.7.4 Report Generation

SQL-BackTrack provides the following reports:

- Backup history

  Detailed information on backup history for physical backups or exports. You can generate a backup history report for the complete database or a subset of the database files.

- Backup priority

  Displays the files that need to be backed up in order of priority, where 1 is less urgent and 100 is most urgent

- Backup generation

  Displays backups that will expire on the next expiration process

- Missing archive logs

  Displays any archive log files missing from the backup

9.2.7.5 Script Creation

With SQL-BackTrack you can back up, restore, and perform other operations interactively. Scripts can also be generated for these operations. These scripts are useful for automating the process and can be scheduled through the ADSM client scheduler. The scripts are stored in the SQL-BackTrack catalog directory.

9.3 Installation and Configuration

SQL-BackTrack for Oracle Version 3.0.0 and 3.0.1 supports the following operating systems:

- AIX 3.2 and above
- SunOS 4.1.x and Solaris 2.2 and above
- HP-UX 9.0 and above
- Digital UNIX 3.2 and 4.0
- SINIX 5.42
- Sequent PTX 4.2.1
- Windows NT 3.51 and 4.0

This section covers SQL-BackTrack installation and configuration on AIX for use with Oracle7 and Oracle8 databases. Oracle 7.2 on AIX requires SQL-BackTrack for Oracle Version 3.0.0. Oracle 7.3 and above and Oracle 8 on AIX requires SQL-BackTrack for Oracle Version 3.0.1. If you use multiple versions of Oracle, you must install the correct version of SQL-BackTrack for each database. The SQL-BackTrack we used for testing was 3.0.1 on AIX 4.1.4 with Oracle Version 7.3.4 and Oracle 8.0.3.

To install SQL-BackTrack you also need:

- 26 MB of free disk space
- An SQL-BackTrack serial number and license key

For full installation details, see *SQL-BackTrack for Oracle Installation Notes Version 3.0.x.*

9.3.1 Installation

Follow these steps to install the SQL-BackTrack:

1. As the root user, mount the SQL-BackTrack CD-ROM
2. Create the installation directory and change the ownership to the oracle user that will install and use SQL-BackTrack:

```
# mkdir /usr/datatools
# chown oracle:dba /usr/datatools
```

   The recommendation location is /usr/datatools.

3. Switch to the oracle user and set the **DTBASE** environment variable to the installation directory:

```
# su - oracle
% setenv DTBASE /usr/datatools
```

DTBASE Environment Variable

The above example sets the DTBASE environment variable only for the installation process. This variable is key to the normal operation of SQL-BackTrack. This variable also should be set in the login profile for the user that will use SQL-BackTrack.

4. Run the install program from the cdrom:

```
% cd /cdrom
% ./install
```

   The install program extracts the files to the $DTBASE directory and then issues a series of prompts for information such as the SQL-BackTrack level to be installed, location of the $DTBASE directory, serial number, and license key information. After you have entered the required information, the installation completes.

At this point SQL-BackTrack is installed and can now be configured.

9.3.2 Configuration

Make sure that the $ORACLE_HOME and $ORACLE_SID environment variables are set for Oracle. Update the PATH variable to include the $DTBASE/obacktrack/bin directory. This directory contains the **obacktrack** command, which is the main SQL-BackTrack executable.

9.3.2.1 Update Database User Privileges

The user that will run SQL-BackTrack must have the privileges for backup recovery operation on the database. Any user with a DBA role has all of the necessary privileges.

SQL-BackTrack provides an SQL script to create a special DTOBACKUP role which has the minimum set of privileges necessary for backup and recovery. You can run this script to create the role in the database and then assign this role to the Oracle user performing the backup. The following steps create and assign the role:

1. Log on to the oracle account.
2. Change to the $DTBASE/obacktrack/bin directory.
3. Start the svrmgrl utility, connect as the internal user, and run the dtobackup.sql script:

```
% svrmgrl
   ⋮
SVRMGR> connect internal
connected
SVRMGR> @dtobackup
Statement processed.
⋮
```

   This macro creates a role named DTOBACKUP in the database.

4. Grant this role to any database user that will run backup and recovery with SQL-BackTrack.

9.3.2.2 Create the SQL-BackTrack Catalog

The first time the obacktrack program is executed, SQL-BackTrack asks whether you want to create the catalog. Answering yes to this question creates the catalog:

```
%./obacktrack
Enter BackTrack catalog directory [/usr/datatools/oracatalog]
No database catalog exists for '/usr/datatools/oracatalog'; create one? (y/n) [y]
SQL-BackTrack for Oracle (OBACKTRACK)
Copyright (c) 1991-1997, DataTools, Inc.
Version 3.0.1, serial number **************
Licensed to: IBM ITSO
(AIX severn 1 4 000091754200)

*** started on 04/17/1998 16:31:48
*** using catalog directory '/usr/datatools/oracatalog'
```

The catalog is created under the $DT_ORACATALOG directory.

After the catalog is created, the main menu of SQL-BackTrack is displayed:

```
SQL-BackTrack for Oracle (OBACKTRACK)
  ⋮
   OBACKTRACK Main Menu

     1. BackTrack catalog administration
     2. Back up (or export) a database
     3. Recover (or import) a database
     4. Logical extraction
     5. Report generation

 Choose: 1-5 c=cancel, q=quit, ?=help
OBACKTRACK>
```

SQL-BackTrack is now configured and can be used. However, only the default disk and tape OBSI modules are installed. To use ADSM the ADSM OBSI module must also be installed.

9.4 ADSM OBSI Module Installation

Install the SQL-BackTrack ADSM OBSI module requires:

- SQL-BackTrack installed and configured
- 4 MB of disk space

Version 2.2 of the SQL-BackTrack ADSM OBSI module supports ADSM Versions 2 and 3. However, the ADSM API is no longer shipped with the ADSM OBSI module. Version 2.1.0.6 or later of the ADSM API client must be installed before you install the ADSM OBSI module.

To install the ADSM OBSI module, log on as root and mount the SQL-BackTrack ADSM OBSI CD-ROM. Run the install program from the CD-ROM. This displays an install menu from which OBSI ADSM should be selected:

```
%./install

       The products available for installation are:

 1) OBSI ADSM
 2) OBSI Disk
 3) OBSI Networker
 4) OBSI SDK
 5) OBSI Tape
 q) Quit Installation

Enter menu number of product to be installed [1-5 | q] : 1
```

The install program prompts for whether this is a new installation and to confirm the $DTBASE directory. Following these prompts the installation unpacks the ADSM OBSI module and displays the following on completion:

```
  ⋮
****** Updating ADSM OBSI links in /usr/datatools *********



***** Installation COMPLETED. Make sure that your obsi.adsm
***** link in your SQL-BackTrack 'links' directory points to
***** /usr/datatools/obsi.adsm

***** Please complete any manual installation procedures as
***** described in the ADSM OBSI Module Version 2.2.1
***** Installation Notes.
```

After the installation completes, the appropriate symbolic links must be verified and created. The ADSM OBSI installation creates a symbolic link in the $DTBASE directory of the Version 2.2.1 ADSM OBSI module:

```
% cd $DTBASE
% ls -l obsi.adsm
lrwxrwxrwx   1 oracle   dba         15 Apr 20 14:04 obsi.adsm -> obsi.adsm-2.2.1
```

The SQL-BackTrack programs must next be linked to the obsi.adsm module:

```
% cd $DTBASE/obacktrack/links
# ln -s ../../obsi.adsm obsi.adsm
```

The ADSM OBSI module is now ready to be used with SQL-BackTrack. However, before it can be used, the ADSM API client must be configured.

9.4.1 Configure ADSM API Client

SQL-BackTrack uses the ADSM API client to communicate with an ADSM server. This client is configured in a similar manner to other ADSM clients, using the dsm.sys and dsm.opt files. You can use the dsm.opt and dsm.sys files that are installed with the ADSM API client or create unique options files for the use of SQL-BackTrack. As a minimum, communication options defining the network address must be defined in the dsm.sys file.

The ADSM API uses environment variables to locate the options files. Therefore different options files can be used for API applications from those used by the backup-archive client. These environment variables must be defined in the oracle user′s environment so that SQL-BackTrack can locate the ADSM API options files. The syntax for environment variable definition is:

ksh: export VAR_NAME=value

```
export DSMI_CONFIG=/home/yourlogin/dsm.opt
```

csh/tcsh: setenv VAR_NAME value

```
setenv DSMI_CONFIG /home/yourlogin/dsm.opt
```

The following environment variables must be set for the API:

DSMI_CONFIG Fully qualified name that points to dsm.opt.

DSMI_DIR Fully qualified name that points to the directory that contains dsm.sys.

DSMI_LOG Fully qualified name that points to the directory that contains the API error log file (dsierror.log). For error log files we recommend that you create a directory where you want the error log to be held and let DSMI_LOG point to that directory.

These three variables should be added to the login profile for the oracle user and any other users that will use SQL-BackTrack

9.4.1.1 Customize Management Class

As you can use both backup-archive and API clients on the same node, you should consider implementing separate policies for the different types of data. Therefore it is recommended that a separate management class be defined for SQL-BackTrack backups.

SQL-BackTrack determines the names of the objects that it stores on the ADSM server. Each object receives a unique numerical name, so each successive backup creates a different object name. Therefore, only one copy of an object is ever stored on the ADSM server. For this reason the following backup copygroup options should be defined for the SQL-BackTrack management class:

```
VEREXIST=1
VERDELETE=0
RETONLY=0
```

SQL-BackTrack deletes old backup objects from the ADSM server by issuing a delete object API call to change the status of the object from active to inactive on the ADSM server. When SQL-BackTrack inactivates old objects on the ADSM server, the above copygroup options will result in these inactive objects being deleted the next time an expiration process is run on the ADSM server.

When you back up the database, the default management class for the node name is used unless you override it with a different management class. With SQL-BackTrack you can define the management class in the following ways:

- Choose the management class to be used when defining the ADSM backup pool in SQL-BackTrack. This is the recommended way (see 9.5, “SQL-BackTrack Catalog” on page 244).

- Add to dsm.sys the INCLEXCL option pointing to your client include-exclude list. Then add the following statement in the include-exclude file:

  ```
  INCLUDE /backtrack*/.../* ORACLE
  ```

This statement rebinds the SQL-BackTrack backup with the filespace name /backtrack* to the ORACLE management class.

9.4.1.2 Update Server COMMTIMEOUT

During certain SQL-BackTrack operations there can be a long delay where there is no apparent ADSM activity. If such an operation exceeds the ADSM COMMTIMEOUT value, ADSM terminates the SQL-BackTrack backup or recovery. The COMMTIMEOUT value should be set to at least 600 sec (10 min) and is defined in the dsmserv.opt server options file.

9.4.1.3 Setting Client Password

It is recommended that you set the PASSWORDACCESS GENERATE option to handle the ADSM API client password. If you use the password generate option, you are not prompted for the API client password while performing SQL-BackTrack operations.

When setting passwordaccess to generate, you must run the adsmpw program. This program is installed with the SQL-BackTrack ADSM OBSI module in the $DTBASE/obsi.adsm/bin directory. The adsmpw program must be run as root. You are prompted for the current password and new password:

```
# cd /usr/datatools/obsi.adsm/bin
# export DSMI_CONFIG=/home/oracle/dsm.opt
# export DSMI_DIR=/usr/lpp/adsmagent/aob
# ./adsmpw

SQL-BackTrack ADSM Module Password Program

Enter current password: ******
Enter new password: ******
Enter new password again: ******
Password changed
#
```

A session is started with the ADSM server, and the password is validated. You can use adsmpw to change the client password at any time.

Environment Variables

The adsmpw program must be run as the root user, and it requires that the DSMI_CONFIG and DSMI_DIR environment variables be set. In the example above these variables were not set in the root user's profile, so they were manually set by using the export command before adsmpw was run.

9.4.2 Validate Installation

You can ensure that the ADSM OBSI module is installed properly by performing the following test:

1. Change to the $DTBASE/obsi.adsm/bin directory and validate the DSMI_DIR environment variable with the following command:

```
% adsmrc 0
0: ANS0302I (RC0)    Successfully done.
```

2. Run the obsitest command. This command writes data to and from the ADSM OBSI module, validating that it is working correctly. First it is used to create a pool entry for the ADSM OBSI module:

```
% obsitest -msgfile ../msgs/obsitest.english obsi/adsm pool
```

This command writes an output file under the obsi.adsm directory:

```
% obsitest -msgfile ../msgs/obsitest.english obsi/adsm out
```

This command reads back the file generated on the previous command.

```
% obsitest -msgfile ../msgs/obsitest.english obsi/adsm in
```

3. Check the ADSM OBSI module by creating a backup pool. Enter the **obacktrack** command to display the main menu:

```
SQL-BackTrack for Oracle (OBACKTRACK)
Copyright (c) 1991-1997, DataTools, Inc.
Version 3.0.1, serial number OB-12345678-2866
Licensed to: itso sj
(AIX bering 1 4 000027463400)

*** started on 02/24/1998 19:04:58
*** using catalog directory '/home/oracle/sqlcat'
   ⋮
Press RETURN to continue...
   OBACKTRACK Main Menu

     1. BackTrack catalog administration
     2. Back up (or export) a database
     3. Recover (or import) a database
     4. Logical extraction
     5. Report generation

 Choose: 1-5 [c=cancel, q=quit, ?=help]
OBACKTRACK> 1
```

Choose the **BackTrack catalog administration** option.

```
   Database Catalog Options

     1. Create/modify a physical profile
     2. Create/modify an export profile
     3. Create a new backup pool or poolgroup
     4. Synchronize database
     5. Expire backups

 Choose: 1-5 c=cancel, q=quit, ?=help“
OBACKTRACK> 3
```

Choose the **Create a new backup pool or poolgroup** option. You are prompted with a series of questions. The first two questions refer to the backup pool name and the OBSI device to use. Make sure that the ADSM OBSI module exists:

```
Backup pools and poolgroups defined in this profile directory:

Enter backup pool or poolgroup name:

 Available OBSI backup devices:

 adsm, disk, tape, poolgroup

 Enter the OBSI backup device type:
```

If ADSM is not on the list of the available OBSI backup devices, the ADSM OBSI module is not properly installed. If all of the previous test steps were completed successfully, check all your links again.

9.5 SQL-BackTrack Catalog

The recovery catalog contains backup configuration and history information for all databases on the machine.

In this section we describe the structure of the catalog, how to configure the catalog, and how to back up the catalog.

9.5.1 SQL-BackTrack Catalog Structure

The catalog is located in the **$DT_ORACATALOG** directory. Figure 116 provides an overview of the catalog directory structure.

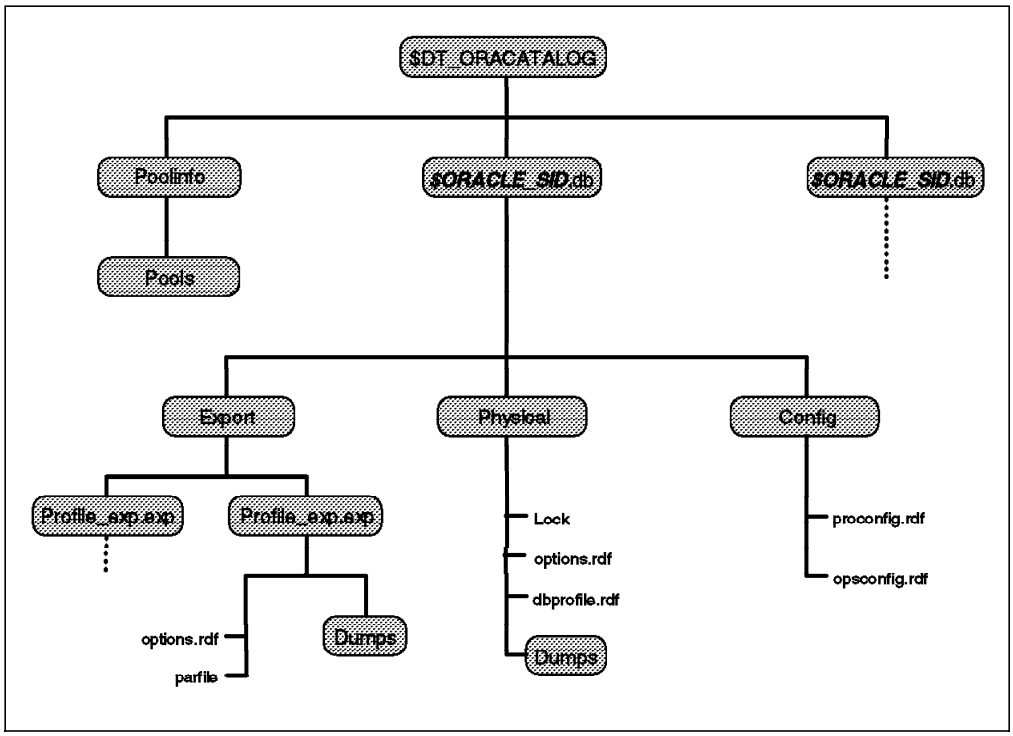


Figure 116. SQL-BackTrack Catalog Structure

Multiple Oracle instances can be defined in the catalog by using the instances SID value to form a subdirectory named SID.db for the instance.

The poolinfo directory contains one file, POOLS, which defines the backup pools and backup groups.

For each database on the machine there is a database subdirectory called *$ORACLE_SID.db*. Each database subdirectory contains the directories for physical and export backups. The subdirectory also contains the config subdirectory for OPS installations. There are two more subdirectories, purge and restart, for internal processes.

The physical subdirectory contains the physical profile information. There is one physical subdirectory in every database directory. The physical subdirectory contains:

- **dbprofile.rdf**: contains the database structure and the profile information
- **options.rdf**: contains general physical backup options
- **lock file**: used internally during backup and recovery
- **dumps directory**: contains backup history information

The **export** subdirectory under the database directory contains a subdirectory for each export profile. The name of the subdirectory is the name of the export profile with the extension .exp. Each export directory contains two files:

- **options.rdf**: contains general export options
- **parfile**: contains parameters for the Oracle exp utility

9.5.2 SQL-BackTrack Catalog Configuration

Before performing any backups and restores with SQL-BackTrack, you must first define a backup pool and physical backup and export profiles.

9.5.2.1 Define Backup Pool

The SQL-BackTrack backup pool contains information about the physical device you will use to store the backup. This process creates the POOL file within the poolinfo subdirectory. The information includes the type of the device and the appropriate options for this device. You can define several backup pools for different kinds of backups. After you have created the backup pool, it can be used for either backup or export.

Follow these steps to define a backup pool:

1. Issue the **obacktrack** command.
2. From the main menu, choose the **BackTrack catalog administration** option:

```
SQL-BackTrack for Oracle (OBACKTRACK)
   ⋮
   OBACKTRACK Main Menu

     1. BackTrack catalog administration
     2. Back up (or export) a database
     3. Recover (or import) a database
     4. Logical extraction
     5. Report generation

 Choose: 1-5 c=cancel, q=quit, ?=help
OBACKTRACK> 1
```

3. Choose the **Create a new backup pool or poolgroup** option:

```
   Database Catalog Options

     1. Create/modify a physical profile
     2. Create/modify an export profile
     3. Create a new backup pool or poolgroup
     4. Synchronize database
     5. Expire backups

 Choose: 1-5 c=cancel, q=quit, ?=help
OBACKTRACK> 3
```

4. You are prompted with a series of questions:

```
Backup pools and poolgroups defined in this profile directory:

Enter backup pool or poolgroup name: ADSM_POOL

Available OBSI backup devices:

  adsm, disk, tape, poolgroup

Enter the OBSI backup device type: adsm

Unless you will be using this OBSI with an OPS database,
the answer to the following question should be no.

Run OBSI on node other than local node? (y/n) (n)

Enter ADSM session node name (return = default)?

Enter ADSM client password (return = none)?

Wait for media mounts? (y/n) (y)

Enter ADSM filespace name (return = /BACKTRACK)?

Enter ADSM management class override(return = none)? oracle

Select ADSM storage pool BACKUP|ARCHIVE (return = BACKUP)?
```

5. After you answer all the questions, you get a summary of the options selected:

```
---------------- Backup Pool Information ----------------

begin backup_pool=ADSM_POOL
  obsi=adsm
  mount-wait=y
  filespace=/BACKTRACK
  management_class=oracle
  storage-pool=backup
  obsi_protocol=3.0
end backup_pool

-------------- End Backup Pool Information --------------

Is this info correct? (y/n) yes

The poolfile has been updated to reflect your changes.
```

These are the stanzas that will be written to the POOL file under the $DT_ORACATALOG/poolinfo directory.

9.5.2.2 Create Physical Profile

The **physical profile** contains the options to use while performing backups. You have to create the physical profile before you can run any physical backups. This process creates the physical directory and the configuration files and scans the database for the structure and incarnation. The physical profile is the base for all of the database backups, and you have to define it only once for each database.

You are prompted with a series of questions during the profile configuration. One of the questions is which **backup pool** to use for the physical backups. If you do not define the backup pool before creating the profile, you are prompted to create one during the profile configuration process.

Following these steps to create a physical profile:

1. Issue the **obacktrack** command and from the main menu choose the **BackTrack catalog administration** option.
2. This displays the Database Catalog Options menu. Choose the **Create/modify a physical profile** option:

```
   Database Catalog Options

     1. Create/modify a physical profile
     2. Create/modify an export profile
     3. Create a new backup pool or poolgroup
     4. Synchronize database
     5. Expire backups

 Choose: 1-5 c=cancel, q=quit, ?=help
OBACKTRACK>
```

3. You are prompted with a series of questions:

```
Enter Oracle SID of target database (return=DFM7):
Enter Oracle HOME directory for DFM7 (return=/u01/app/oracle/product/734):
Log in as user with one of these roles: DBA, DTOBACKUP
Enter Oracle username for database 'DFM7': system
Enter password for user 'system':
Do you want to store user 'system' for backup?  (y/n) (y)
Do you want to compress backed up data? (y/n) (y)
Do you want to encrypt backed up data? (y/n) (n)
Enter minimum backup expiration value in days (unlimited):
Enter minimum number of backups to keep (1):
Auto-expire dump history after each backup? (y/n) (y)
Enable physical incremental backups(y/n) (y)
Maximum archived logs to backup per set (100):
RENAME or DELETE archive log files after backup (rename)?
Monitor archive log destination?  (y/n) (n)
Enter log_arch_dest utilization threshold (80):
Enter archive log file count threshold (0):
Enter DTOARCHMON interval for checking archive logs (3600):
Always backup init.ora files? (y/n) (y)
Always backup control files? (y/n) (y)
Enable short BEGIN/END BACKUP window for online backups? (y/n) (y)
Backup pools and poolgroups defined in this profile directory:
   [Pool]  ADSM_POOL
Enter new or existing backup pool or poolgroup name : ADSM_POOL
Enable logical extraction? (y/n) (y)
Use a different backup pool for logical data? (y/n) (n)
Creating database incarnation #1.
Adding datafile '/u01/oradata/DFM7/system01.dbf' to catalog...
Adding datafile '/u01/oradata/DFM7/rbs01.dbf' to catalog...
Adding datafile '/u01/oradata/DFM7/temp01.dbf' to catalog...
Adding datafile '/u01/oradata/DFM7/tools01.dbf' to catalog...
Adding datafile '/u01/oradata/DFM7/users01.dbf' to catalog...
Adding datafile '/u01/oradata/DFM7/test1.dbf' to catalog...
Adding datafile '/u01/oradata/DFM7/test2.dbf' to catalog...
Adding datafile '/u01/oradata/DFM7/test3.dbf' to catalog...
Adding datafile '/u01/app/oracle/product/734/dbs/georgetest01.dbf' to catalog...

Physical profile for database DFM7 has been created.

Press RETURN to continue...
```

As this is the first time that you create a physical profile, the process also creates the database incarnation.

9.5.2.3 Create Export Profile

The **export profile** contains information about every export operation you want to perform. You have to configure an export profile for each type of export you want to do. The process creates subdirectories for each export profile under the export directory.

Follow these steps to create an export profile:

1. Issue the **obacktrack** command.
2. From the main menu choose the **BackTrack catalog administration** option.
3. Choose the **Create/modify an export profile** option:

```
   Database Catalog Options

     1. Create/modify a physical profile
     2. Create/modify an export profile
     3. Create a new backup pool or poolgroup
     4. Synchronize database
     5. Expire backups

 Choose: 1-5 c=cancel, q=quit, ?=help
OBACKTRACK> 2
```

You are prompted with a series of questions. There are three groups of questions:

- Questions about the database and the type of export to perform:

```
Enter Oracle SID of target database (return=DFM7):
Enter Oracle HOME directory for DFM7 return=/u01/app/oracle/product/734  :
Enter Oracle username for database 'DFM7':
Enter password for user 'system':
Enter new export profile name: expDFM7
   Choose type of export

     1. FULL export
     2. USER level export
     3. TABLE level export
     4. COMPLETE, CUMULATIVE, INCREMENTAL level export

 Choose: 1-4 c=cancel, q=quit, ?=help
OBACKTRACK>
```

- Questions about the export operation, such as number of copies and backup pool to be used:

```
Do you want to store user 'system' for backup?  (y/n) y
Do you want to compress backed up data? (y/n) y
Do you want to encrypt backed up data? (y/n) n
Enter minimum backup expiration value in days(unlimited):
Enter minimum number of backups to keep: (1)
Auto-expire dump history after each backup? (y/n)
Backup pools and poolgroups defined in this profile directory:
   Pool  ADSM_POOL
Enter new or existing backup pool name : ADSM_POOL
```

- The export parameters for the defined backup:

```
Enter values for the following export parameters:
Enter insert buffer size (minimum is 4096):(Return = default):
Export grants?  (Y)
Export indexes?  (Y)
Export table data?  (Y)
Export table constraints?  (Y)
Compress table data?  (Y)
Parameter file containing additional options(Return = none):
Log file(Return = none):
Export read-consistent view of database? (N)
Statistics: (ESTIMATE)
Use direct path export?  (N)
Export profile expDFM7 for database DFM7 has been created.
```

Having created a storage pool and physical and export profiles, you are now ready to use SQL-BackTrack.

9.5.3 SQL-BackTrack Catalog Backups

The SQL-BackTrack catalog contains all of the configuration information that is essential for database backup and recovery. The catalog directory tree is updated after each backup. You should regularly back up the SQL-BackTrack catalog.

As the catalog is a collection of files and directories, we recommend that you use the ADSM backup-archive client to back it up. You can either schedule a selective backup of the directory after the database backup has completed or run the **dsmc selective** command at the end of your backup script.

The following is an example of backing up the SQL-BackTrack catalog directory ($DT_ORACATALOG) by using the ADSM selective backup command:

```
% echo $DT_ORACATALOG
% /home/oracle/sqlcat
% dsmc selective /home/oracle/sqlcat -subdir=yes
%
```

9.6 Backup and Recovery Examples

This section describes some typical uses of SQL-BackTrack. The following examples illustrate the concepts of using SQL-BackTrack:

- Tablespace backup
- Database export
- Restore of a data file
- Recover an individual table through logical table extraction

For detailed information refer to the *SQL-BackTrack for Oracle User's Guide*.

For these examples Oracle Version 7.3.4 on AIX 4.1.2 was used.

9.6.1 Tablespace Backup

This example shows you how to perform a tablespace backup. The example includes:

- Backup script generation
- Backup dry run
- Actual backup

1. Choose the **Back up (or export) a database** option from the main menu:

```
SQL-BackTrack for Oracle (OBACKTRACK)

     1. BackTrack catalog administration
     2. Back up (or export) a database
     3. Recover (or import) a database
     4. Logical extraction
     5. Report generation

 Choose: 1-5 c=cancel, q=quit, ?=help
OBACKTRACK> 2
```

2. Select the database you want to back up:

```
   Select a database

     1. DFM7
     2. MDG8

 Choose: 1-2 c=cancel, q=quit, ?=help
OBACKTRACK> 1
```

3. Choose the backup type (in our case, a Physical backup):

```
   Choose backup type

     1. Physical backup
     2. Export

 Choose: 1-2 c=cancel, q=quit, ?=help
OBACKTRACK> 1
```

4. Choose the type of physical backup (in our case, a tablespace backup):

```
   Choose type of backup

     1. Back up the entire database
     2. Back up selected tablespaces
     3. Back up selected data files
     4. Back up archived redo log files

 Choose: 1-4 c=cancel, q=quit, ?=help
OBACKTRACK> 2
```

5. Choose the tablespace to back up by entering the tablespace number:

```
   Select tablespaces to back up:

     1. GEORGETEST                                                  ( )
     2. RBS                                                         ( )
     3. SYSTEM                                                      ( )
     4. TEMP                                                        ( )
     5. TESTMUX                                                     ( )
     6. TOOLS                                                       ( )
     7. USERS                                                       ( )

 Choose: 1-7 a=accept, c=cancel, q=quit, ?=help
OBACKTRACK> 6 7
```

6. Confirm the choices made:

```
   Select tablespaces to back up:

     1. GEORGETEST                                                  ( )
     2. RBS                                                         ( )
     3. SYSTEM                                                      ( )
     4. TEMP                                                        ( )
     5. TESTMUX                                                     ( )
     6. TOOLS                                                       (x)
     7. USERS                                                       (x)

 Choose: 1-7 a=accept, c=cancel, q=quit, ?=help
OBACKTRACK> a
```

7. Choose the type of backup:

```
ONLINE or COLD backup?  (ONLINE)
(F)ull or (I)ncremental?  (FULL)
```

You can choose one of the following options:

```
   Backup Menu

     1. Start backup
     2. Start dryrun backup
     3. Generate backup script

 Choose:
1-3 c=cancel, q=quit, ?=help
```

In this example we first generate a script, then run a dry-run backup, and finally run the backup. Figure 117 is an example of the script generated by selecting **3** from the Backup Menu.

```
OBACKTRACK> #!/bin/sh
#%DT% SQL-BackTrack <DO NOT REMOVE OR CHANGE THIS LINE>

# SQL-BackTrack for Oracle - Copyright (c) 1995-1997
#
# This is a generated script; call this script from cron or
# from other schedulers to perform the requested operation at regular
# intervals.

DTBASE=/usr/datatools; export DTBASE

# cd to a known directory...
cd /usr/datatools

/usr/datatools/obacktrack-3.0.1/bin/dtobackup /home/oracle/sqlcat -database DFM7
-tablespace TOOLS -tablespace USERS -online -full -quiet -noprompt
status=$?

# Be sure to exit the script with the exit status.  If you add
# anything to the script after the command you should be sure
# to exit with the correct status.
exit $status

Press RETURN to continue...Enter file name:  (Return = none)
Saved in '/usr/datatools/sqllog/tables1'.
```

Figure 117. SQL-BackTrack Sample Tablespace Backup Script

You can use this script to run the backup with the ADSM scheduler. The script can be part of the backup operation or a separate script.

A dry-run backup is performed by selecting **2** from the Backup Menu. Figure 118 is the output of the dry-run backup.

```
OBACKTRACK>
 dtocheck 35830  Verifying options for DFM7 -- physical.
 dtocheck 35830  Resyncing database with profile...
 dtobackup 41966  09:39 DRYRUN: backup of tablespace TOOLS dictionary
 completed successfully
 dtobackup 41966  09:39 DRYRUN: backup of tablespace USERS dictionary
 completed successfully
 dtobackup 41966  09:39 DRYRUN: backup of control files completed successfully
 dtobackup 41966  09:40 DRYRUN: backup of /u01/oradata/DFM7/tools01.dbf
 completed successfully
 dtobackup 41966  09:40 DRYRUN: backup of /u01/oradata/DFM7/users01.dbf
 completed successfully
 dtobackup 41966  09:41 DRYRUN: backup of parameter files completed successfully
 DTOBACKUp 41966  09:41 DRYRUN: backup of archived logs completed successfully
```

Figure 118. SQL-BackTrack Dry-run Backup Output

The dry run performs a test of the backup, but no actual data is sent to the ADSM server.

The actual backup is performed by selecting **1** from the Backup menu. Figure 119 shows the output of the backup operation.

```
OBACKTRACK>
 dtocheck 38754  Verifying options for DFM7 -- physical.
 dtocheck 38754  Resyncing database with profile...
 dtobackup 29508  09:44 backup of tablespace TOOLS dictionary completed
 dtobackup 29508  09:45 backup of tablespace USERS dictionary completed
 dtobackup 29508  09:45 backup of control files completed
 dtobackup 29508  09:45 backup of /u01/oradata/DFM7/tools01.dbf completed
 dtobackup 29508  09:45 backup of /u01/oradata/DFM7/users01.dbf completed
 dtobackup 29508  09:46 backup of parameter files completed
 dtobackup 29508  09:47 backup of archived logs completed
 dtobackup 29508  09:47 backup of archived logs completed
```

Figure 119. SQL-BackTrack Tablespace Physical Backup Output

9.6.2 Database Export

This example shows you how to perform an export with SQL-BackTrack. The example includes:

- Export script generation
- Export operation

1. Choose the **Back up (or export) a database** option from the main menu:

```
SQL-BackTrack for Oracle (OBACKTRACK)

     1. BackTrack catalog administration
     2. Back up (or export) a database
     3. Recover (or import) a database
     4. Logical extraction
     5. Report generation

 Choose: 1-5 c=cancel, q=quit, ?=help
OBACKTRACK> 2
```

2. Choose the database you want to export:

```
   Select a database

     1. DFM7
     2. MDG8

 Choose: 1-2 c=cancel, q=quit, ?=help
OBACKTRACK> 1
```

3. Choose the backup type (in this case, an export):

```
   Choose backup type

     1. Physical backup
     2. Export

 Choose: 1-2 c=cancel, q=quit, ?=help
OBACKTRACK> 2
```

4. Choose the export profile (in this example we run an export at the user level (scott)):

```
   Choose an export profile

     1. expDFM7
     2. scott_exp

 Choose: 1-2 c=cancel, q=quit, ?=help
OBACKTRACK> 2
```

You can choose one of the following options:

```
   Export Menu

     1. Start export
     2. Start dryrun export
     3. Generate export script
```

In this example we create an export script and run the export. Figure 120 is an example of the export script generated by selecting **3** from the Export Menu.

```
OBACKTRACK> #!/bin/sh
#%DT% SQL-BackTrack <DO NOT REMOVE OR CHANGE THIS LINE>

# SQL-BackTrack for Oracle - Copyright (c) 1995-1997
#
# This is a generated script; call this script from cron or
# from other schedulers to perform the requested operation at regular
# intervals.

DTBASE=/usr/datatools; export DTBASE

# cd to a known directory...
cd /usr/datatools

/usr/datatools/obacktrack-3.0.1/bin/dtobackup  /home/oracle/sqlcat
-database DFM7 -export scott_exp -quiet -noprompt
status=$?

# Be sure to exit the script with the exit status.  If you add
# anything to the script after the command you should be sure
# to exit with the correct status.
exit $status

Press RETURN to continue...
Enter file name:  (Return = none) scott_exp
Saved in '/usr/datatools/sqllog/export/scott_exp'.
```

Figure 120. SQL-BackTrack Sample Export Script

You can use this script to run the export with the ADSM scheduler. The script can be a part of the backup operation or a separate script.

Figure 121 on page 255 is the output of the export operation run by selecting **1** from the Export Menu.

```
OBACKTRACK>  dtocheck 38132  Verifying options for DFM7/scott_exp export.
(dtoexport 37918) Forking export process
(dtoexport 37918) /u01/app/oracle/product/734/bin/exp system COMPRESS=Y
CONSISTENT=N CONSTRAINTS=Y DIRECT=N
FILE=/tmp/datatools/dtoexport.37918-0.26.02.1998.16:31:51.f
GRANTS=Y INDEXES=Y OWNER=scott ROWS=Y STATISTICS=ESTIMATE
(dtobackup 15602) (dtoexport 37918) Export: Release 7.3.4.0.0 -
Production on Thu Feb 26 16:31:52 1998
(dtobackup 15602) (dtoexport 37918) Copyright (c) Oracle Corporation 1979,
1996.  All rights reserved.
(dtobackup 15602) (dtoexport 37918) Connected to:
Oracle7 Server Release 7.3.4.0.0 - Production
(dtobackup 15602) (dtoexport 37918) PL/SQL Release 2.3.4.0.0 - Production
(dtobackup 15602) (dtoexport 37918) Export done in US7ASCII character set
(dtobackup 15602) (dtoexport 37918) About to export specified users ...
(dtobackup 15602) (dtoexport 37918) About to export SCOTT's objects ...
(dtobackup 15602) (dtoexport 37918) . exporting snapshots
(dtobackup 15602) (dtoexport 37918) . exporting snapshot logs
(dtobackup 15602) (dtoexport 37918) . exporting job queues
(dtobackup 15602) (dtoexport 37918) . exporting refresh groups and children
(dtobackup 15602) (dtoexport 37918) . exporting database links
(dtobackup 15602) (dtoexport 37918) . exporting sequence numbers
(dtobackup 15602) (dtoexport 37918) . exporting cluster definitions
(dtobackup 15602) (dtoexport 37918) . about to export SCOTT's tables via Conventional Path ...
(dtobackup 15602) (dtoexport 37918) . . exporting table BONUS 0 rows exported
(dtobackup 15602) (dtoexport 37918) . . exporting table DEPT  4 rows exported
(dtobackup 15602) (dtoexport 37918) . . exporting table DUMMY 1 rows exported
(dtobackup 15602) (dtoexport 37918) . . exporting table EMP   14 rows exported
(dtobackup 15602) (dtoexport 37918) . . exporting table SALGRADE 5 rows exported
(dtobackup 15602) (dtoexport 37918) . exporting synonyms
(dtobackup 15602) (dtoexport 37918) . exporting views
(dtobackup 15602) (dtoexport 37918) . exporting stored procedures
(dtobackup 15602) (dtoexport 37918) . exporting referential integrity constraints
(dtobackup 15602) (dtoexport 37918) . exporting triggers
(dtoexport 42810) Wrote 4K bytes to export dump stream.
(dtobackup 15602) (dtoexport 37918) . exporting posttables actions
(dtobackup 15602) (dtoexport 37918) Export terminated successfully without warnings.
(dtoexport 37918) *** finished on 02/26/1998 16:31:58
(dtoexport 37918) *** highest return code was 0
(dtoexport 37918) *** export operation completed successfully
Export completed successfully.
```

Figure 121. SQL-BackTrack Export Output

9.6.3 Restore of a Data File

This example shows you how to perform a restore of a data file with SQL-BackTrack. In this example we restore one data file within a tablespace.

1. From the main menu choose the **Recover (or import) a database** option:

```
     1. BackTrack catalog administration
     2. Back up (or export) a database
     3. Recover (or import) a database
     4. Logical extraction
     5. Report generation

 Choose: 1-5 c=cancel, q=quit, ?=help
OBACKTRACK> 3
```

2. Choose the database to recover:

```
   Select a database

     1. DFM7
     2. MDG8

 Choose: 1-2 c=cancel, q=quit, ?=help
OBACKTRACK> 1
```

3. Choose the recovery type:

```
   Choose recover type

     1. Physical recovery
     2. Import

 Choose: 1-2 c=cancel, q=quit, ?=help
OBACKTRACK> 1
```

4. Choose the recovery method:

```
   OBACKTRACK Database File Restore Menu

     1. Restore selected database files
     2. Perform a time-based recovery
     3. Perform a change-based recovery

 Choose: 1-3  c=cancel, q=quit, ?=help
OBACKTRACK> 1
```

5. Choose the data file to recover:

```
   OBACKTRACK Recover File Selection Menu

     1. Init.ora file: /u01/app/oracle/admin/DFM7/pfile/configDFM7.ora ( )
     2. Init.ora file: /u01/app/oracle/product/734/dbs/initDFM7.ora    ( )
     3. Redolog: /u01/oradata/DFM7/redoDFM701.log                      ( )
     4. Redolog: /u01/oradata/DFM7/redoDFM702.log                      ( )
     5. Redolog: /u01/oradata/DFM7/redoDFM703.log                      ( )
     6. Tablespace RBS: /u01/oradata/DFM7/rbs01.dbf                    ( )
     7. Tablespace SYSTEM: /u01/oradata/DFM7/system01.dbf              ( )
     8. Tablespace TEMP: /u01/oradata/DFM7/temp01.dbf                  ( )
     9. Tablespace TESTMUX: /u01/oradata/DFM7/test1.dbf                ( )
    10. Tablespace TESTMUX: /u01/oradata/DFM7/test2.dbf                ( )
    11. Tablespace TESTMUX: /u01/oradata/DFM7/test3.dbf                ( )
    12. Tablespace TOOLS: /u01/oradata/DFM7/tools01.dbf                ( )
    13. Tablespace USERS: /u01/oradata/DFM7/users01.dbf                ( )

 Choose: 1-13 a=accept, c=cancel, q=quit, ?=help
OBACKTRACK> 13
```

6. Confirm the file chosen:

```
   OBACKTRACK Recover File Selection Menu

     1. Init.ora file: /u01/app/oracle/admin/DFM7/pfile/configDFM7.ora ( )
     2. Init.ora file: /u01/app/oracle/product/734/dbs/initDFM7.ora    ( )
     3. Redolog: /u01/oradata/DFM7/redoDFM701.log                      ( )
     4. Redolog: /u01/oradata/DFM7/redoDFM702.log                      ( )
     5. Redolog: /u01/oradata/DFM7/redoDFM703.log                      ( )
     6. Tablespace RBS: /u01/oradata/DFM7/rbs01.dbf                    ( )
     7. Tablespace SYSTEM: /u01/oradata/DFM7/system01.dbf              ( )
     8. Tablespace TEMP: /u01/oradata/DFM7/temp01.dbf                  ( )
     9. Tablespace TESTMUX: /u01/oradata/DFM7/test1.dbf                ( )
    10. Tablespace TESTMUX: /u01/oradata/DFM7/test2.dbf                ( )
    11. Tablespace TESTMUX: /u01/oradata/DFM7/test3.dbf                ( )
    12. Tablespace TOOLS: /u01/oradata/DFM7/tools01.dbf                ( )
    13. Tablespace USERS: /u01/oradata/DFM7/users01.dbf                (x)

 Choose: 1-13 a=accept, c=cancel, q=quit, ?=help
OBACKTRACK> a
```

7. You are prompted with the following questions:

```
Enter directory for archive logs (/u01/app/oracle/product/734/dbs)
Restore all files to their original locations if possible?(y/n) (y)
File '/u01/oradata/DFM7/users01.dbf'
will be recovered to '/u01/oradata/DFM7/users01.dbf'

Confirm? (y/n) (y)
```

8. Choose one of the following options:

```
   OBACKTRACK Recovery Menu

     1. Start recovery
     2. Start dryrun recovery
     3. Generate recovery procedure

 Choose: 1-3  c=cancel, q=quit, ?=help
OBACKTRACK> 1
```

Figure 122 on page 258 is the output of the restore and recovery operation run by selecting **1** from the Recovery Menu.

```
#  ======================================================================
#  Connect to the database as internal.
#  ======================================================================
CONNECT INTERNAL


#  ======================================================================
#  Take all tablespaces which are being recovered offline.
#  ======================================================================
ALTER TABLESPACE USERS OFFLINE IMMEDIATE;


#  ======================================================================
#  Restore needed data files from backup with the DTORESTORE program.
#  ======================================================================
HOST dtorestore /home/oracle/sqlcat -database DFM7 -resetid 324313535 \
    -datafile /u01/oradata/DFM7/users01.dbf \
    -noinclude -copyover -noconfirm -status
[dtorestore 24078] Starting job #123 on 02/26/1998 18:10:21
[dtorestore 24078] 18:10 restore of /u01/oradata/DFM7/users01.dbf completed


#  ======================================================================
#  Synchronize datafile names between the Oracle restored controlfile
#  and the SQL-BackTrack database catalog.
#  ======================================================================
HOST dtorutil /home/oracle/sqlcat -database DFM7 -syncnames -resetid 324313535



#  ======================================================================
#  Recover individual tablespaces.  obacktrack will automatically
#  restore and apply archived redo logs for this step.
#  ======================================================================
HOST dtorutil /home/oracle/sqlcat -database DFM7 -recover -resetid 324313535
-tablespace USERS


#  ======================================================================
#  Put all tablespaces back online.
#  ======================================================================
ALTER TABLESPACE USERS ONLINE;
Recovery complete!
Resyncing database with profile...
```

Figure 122. SQL-BackTrack Restore Data File Output

9.6.4 Logical Table Extraction

This example shows you how to perform a logical table extraction with SQL-BackTrack. In this example we extract two columns from a table. Logical extraction must be enabled (the default setting) when the physical profile is created. Data can be extracted only from a backup. Exports cannot be used.

1. From the main menu, choose the **Logical extraction** option:

```
SQL-BackTrack for Oracle (OBACKTRACK)

     1. BackTrack catalog administration
     2. Back up (or export) a database
     3. Recover (or import) a database
     4. Logical extraction
     5. Report generation

 Choose: 1-5 c=cancel, q=quit, ?=help
OBACKTRACK> 4
```

2. Select the database from which you want to extract:

```
   Select a database

     1. DFM7
     2. MDG8

 Choose: 1-2 c=cancel, q=quit, ?=help
OBACKTRACK> 1
```

3. Select the tablespace:

```
   Select a tablespace to extract objects from:

     1. GEORGETEST
     2. RBS
     3. SYSTEM
     4. TEMP
     5. TESTMUX
     6. TOOLS
     7. USERS

 Choose: 1-7 c=cancel, q=quit, ?=help
OBACKTRACK> 7
```

4. A series of prompts is used for the required information:

```
Enter a date between '02/25/1998 15:50:26' and '02/26/1998 17:55:05'.

Enter extract time.(return = 02/26/1998 17:55:05)

Extract to time 02/26/1998 17:55:05? (y/n) (y)

Destination directory for extract file? (return = /tmp/datatools)

Working directory for extract file? (return = /tmp/datatools)

Extract format: (S)QL, (F)ixed Tab, (V)ariable, or (L)oad to DB?(SQL)

Date Format?   (return = YYYY/MM/DD HH24:MI:SS)

(dtorestore 24206) Starting job #119 on 02/26/1998 17:55:43

(dtorestore 24206) 17:56 restore of tablespace TOOLS dictionary completed
```

Here the date on which to extract the data is defined, as are the directory to which the data will be extracted and the format of the extracted data. The extract format determines the type of data that will be created:

- A SQL loader file for use with SQL*Plus
- A plain text file
- Loaded directly back into the database

By default an SQL loader file is created. Once you have answered these questions, more menus are displayed.

5. Select the tables to be extracted from the tablespace:

```
   Select Tables to extract:

     1. SCOTT.BONUS                    ( )
     2. SCOTT.DEPT                     ( )
     3. SCOTT.DUMMY                    ( )
     4. SCOTT.EMP                      ( )
     5. SCOTT.SALGRADE                 ( )

 Choose: 1-5 a=accept, c=cancel, q=quit, ?=help
OBACKTRACK> 4
```

6. Accept the chosen table:

```
   Select Tables to extract:

     1. SCOTT.BONUS                    ( )
     2. SCOTT.DEPT                     ( )
     3. SCOTT.DUMMY                    ( )
     4. SCOTT.EMP                      (x)
     5. SCOTT.SALGRADE                 ( )

 Choose: 1-5 a=accept, c=cancel, q=quit, ?=help
OBACKTRACK> a
```

7. Select the columns to be extracted within the table:

```
   Select Column(s) to extract:

     1. SCOTT.EMP   EMPNO              ( )
     2. SCOTT.EMP   ENAME              ( )
     3. SCOTT.EMP   JOB                ( )
     4. SCOTT.EMP   MGR                ( )
     5. SCOTT.EMP   HIREDATE           ( )
     6. SCOTT.EMP   SAL                ( )
     7. SCOTT.EMP   COMM               ( )
     8. SCOTT.EMP   DEPTNO             ( )

 Choose: 1-8 a=accept, c=cancel, q=quit, ?=help
OBACKTRACK> 2 4
```

8. Accept the chosen columns:

```
   Select Column(s) to extract:

     1. SCOTT.EMP   EMPNO              ( )
     2. SCOTT.EMP   ENAME              (x)
     3. SCOTT.EMP   JOB                ( )
     4. SCOTT.EMP   MGR                (x)
     5. SCOTT.EMP   HIREDATE           ( )
     6. SCOTT.EMP   SAL                ( )
     7. SCOTT.EMP   COMM               ( )
     8. SCOTT.EMP   DEPTNO             ( )


 Choose: 1-8 a=accept, c=cancel, q=quit, ?=help
OBACKTRACK>
```

At this point the extraction job runs with the results displayed on the screen (Figure 123 on page 261).

```
OBACKTRACK>
(dtorestore 43176) Starting job #120 on 02/26/1998 17:58:14
(dtorestore 43176) 17:58 restore of /u01/oradata/DFM7/tools01.dbf completed
(dtorestore 43710) Starting job #121 on 02/26/1998 18:02:30
(dtorestore 43710) 18:02 restore of archived logs completed
(dtorestore 43728) Starting job #122 on 02/26/1998 18:02:56
(dtorestore 43728) 18:03 restore of archived logs completed
(dtoextract 41120)    Closing archived redo log file '/tmp/datatools/DTTEMP/arch1_85.dbf'
(dtoextract 41120)    Exported to:     /tmp/datatools/SCOTT.EMP.data.43398.888544488
(dtoextract 41120)    Rows exported:    14
(dtoextract 41120)    Chained rows:     0
(dtoextract 41120)    Migrated rows:    0
(dtoextract 41120)    Dangling rows:    0

OBACKTRACK>
```

Figure 123. SQL-BackTrack Logical Extraction Results

In this example the extracted data has been exported to an SQL loader file, SCOTT.EMP.data.43398.888544488. The first two qualifiers of this filename are the same as the name of the extracted table. This file is formatted so that it can be used with SQL*Plus to load the data back into the database or into another database:

```
CREATE TABLE EMP(EMPNO NUMBER(4), ENAME VARCHAR2(10), JOB VARCHAR2(9),
MGR NUMBER(4), HIREDATE DATE, SAL NUMBER(7,2), COMM NUMBER(7,2), DEPTNO NUMBER(2));


/* 000001DE.0000.0001: */
INSERT INTO EMP(ENAME, MGR)
 VALUES('SMITH', 7902);


/* 000001DE.0009.0001: */
INSERT INTO EMP(ENAME, MGR)
 VALUES('TURNER', 7698);

   ⋮
COMMIT;
```

The INSERT statements contain values only for the ENAME and MGR columns, which are the two columns that were extracted.

Chapter 10. SAP R/3 Oracle and BACKINT/ADSM

This chapter describes how to back up and restore Oracle databases in a SAP R/3 environment with BACKINT/ADSM for Oracle Version 2.2. BACKINT/ADSM is a product that provides an intelligent, transparent interface between the SAP R/3 backup utilities and one or more ADSM servers. BACKINT/ADSM Version 2.2 is supported on AIX, Digital UNIX, HP-UX, NCR, SINIX, Sun Solaris, and Windows NT (Intel-based microprocessor) platforms for SAP/R3 systems using Oracle7 databases.

This chapter is divided into a number of sections. First we introduce SAP R/3 software and its architecture. Then we look at the BACKINT/ADSM structure and explain how BACKINT/ADSM integrates with SAP R/3 and ADSM. We introduce the different backup and restore/recovery types for SAP R/3 Oracle databases and explain the additional BACKINT/ADSM functions provided. We describe the SAP R/3 database administration utilities and commands you can use to back up and restore SAP R/3 Oracle databases. Several examples are presented illustrating some of the methods of backing up and restoring SAP R/3 Oracle databases. We conclude with an overview of how to use ADSM to back up and restore SAP R/3 non-database files and an example of how to back up an entire SAP R/3 system in unattended mode.

This chapter is an introduction to SAP/R3 and BACKINT/ADSM. It is not an authoritative guide to SAP/R3 and Oracle backup and recovery. This chapter should be read in conjunction with the *BACKINT/ADSM Program Description and Operations* manual.

10.1 SAP R/3 and BACKINT/ADSM Introduction

SAP is one of the world's largest software vendors providing packaged commercial applications.

SAP R/3 is an integrated client-server package covering business applications such as accounting, human resources, logistics, and production planning. It works on different hardware and software platforms. The core part of a SAP R/3 installation is the relational database where all business transactions are stored. Therefore backup and recovery of this database is crucial for business availability.

For performance and functional reasons, SAP R/3 introduces a three-tier model where several components communicate with each other through a high-speed LAN or a WAN. This architecture is very flexible and can be scaled up to increase the overall throughput of a SAP R/3 instance.

10.1.1 SAP R/3 Architecture

Figure 124 on page 264 illustrates the SAP R/3 client-server architecture, which consists of database servers, application servers, and presentation clients.

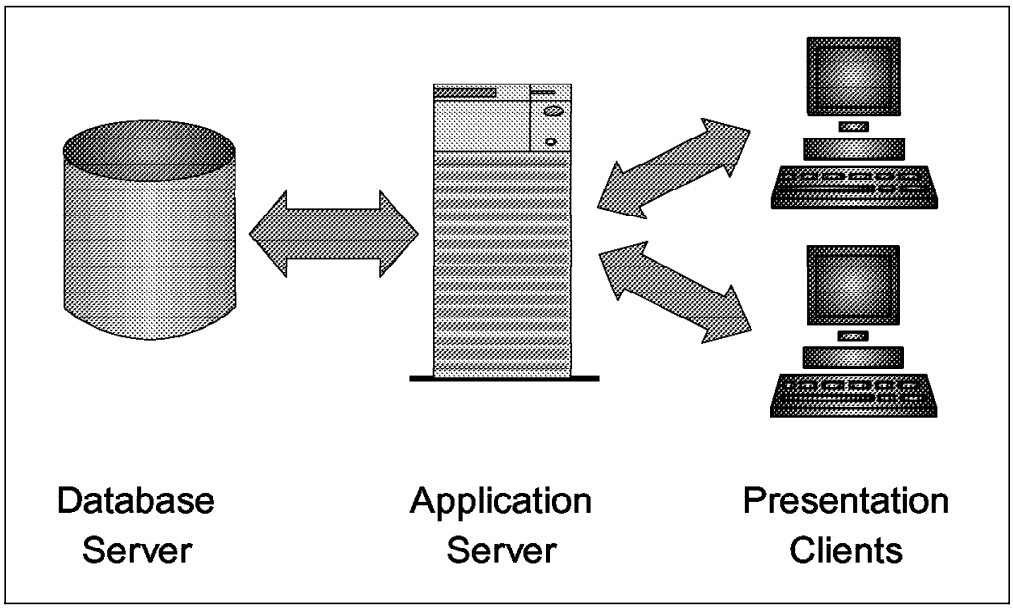


Figure 124. SAP R/3 Client-Server Architecture

A SAP R/3 service can be split so that the database runs on a central machine (database server) and the application runs on a number of separate application servers. Application servers use the code installed on the database server by NFS-mounting the code directories. No data is stored on application servers. The servers are connected through a fast LAN or a high-performance switch such as the RS/6000 SP switch.

The end user is connected to SAP R/3 through a presentation client, typically a PC or an X-terminal. The presentation client uses SAP R/3 transactions to send requests to the application server. The application server reads data from local memory or initiates database requests against the database server, which represents the highest level in the server hierarchy.

The SAP R/3 software, the ABAP/4 application programs, and user data are stored on the database server. The following operating systems are supported:

- R/3 servers
  - UNIX versions of IBM, HP, Sun, DEC, SNI, and NCR
  - Windows NT on Intel platforms
  - OS/390
  - OS/400
- Presentation clients
  - Windows 95
  - Windows 3.1
  - Windows NT
  - OS/2
  - UNIX workstations with Motif
  - Apple Macintosh

A number of different database systems (Oracle, DB2, Informix, and Adabas) are supported for SAP/R3 database servers. This chapter focuses on Oracle.

There are several methods of backing up databases in a SAP R/3 environment, but it is recommended that a method be used that is officially supported by SAP. SAP supports backing up an Oracle database with BACKINT/ADSM. BACKINT/ADSM supports Oracle databases installed on:

- File systems
- Raw devices
- Virtual shared disks on RS/6000 SP

SAP R/3 and EBU

As SAP R/3 uses BACKINT/ADSM and the ADSM API to back up its Oracle databases, it does not use EBU as a backup and restore/recovery solution. The use of EBU for SAP R/3 Oracle databases is not supported by SAP.

10.1.2 SAP R/3 Backup Utilities

SAP R/3 provides its own integrated database administration utility, SAPDBA, which provides a menu-driven interface to the backup and restore utilities, BRBACKUP, BRARCHIVE, and BRRESTORE.

BRBACKUP and BRARCHIVE are commands to back up SAP R/3 Oracle databases. They save the following Oracle files to a target backup device such as an ADSM server:

- Control files
- Online redo log files
- Archived redo log files
- Data files
- Parameter files

The BRBACKUP function of SAPDBA provides full and partial offline and online backups of Oracle databases. BRARCHIVE is used to archive the offline redo logs. BRBACKUP and BRARCHIVE save data either directly to a locally attached tape drive or by connecting to a backup interface such as BACKINT/ADSM. BRRESTORE is used to restore and recover the Oracle database.

10.1.3 BACKINT/ADSM

BACKINT/ADSM is a backup interface between the SAP R/3 backup utilities and one or more ADSM servers. It enables the SAP/R3 backup and recovery functions to transparently use one or more ADSM servers to store backup objects. With BACKINT/ADSM you can use the capabilities of ADSM to establish an enterprisewide consistent backup and restore solution that incorporates the entire SAP R/3 system. BACKINT/ADSM with ADSM provides reliable and repeatable operation procedures with which system administrators can manage large volumes of data more efficiently.

BACKINT/ADSM separates the management of backup storage from normal SAP R/3 operation. It enables SAP R/3 administrators to back up SAP R/3 data using ADSM while using the usual SAP R/3 SAPDBA utilities, BRBACKUP,

BRRESTORE, and BRARCHIVE. Figure 125 on page 266 shows the BACKINT/ADSM architecture.

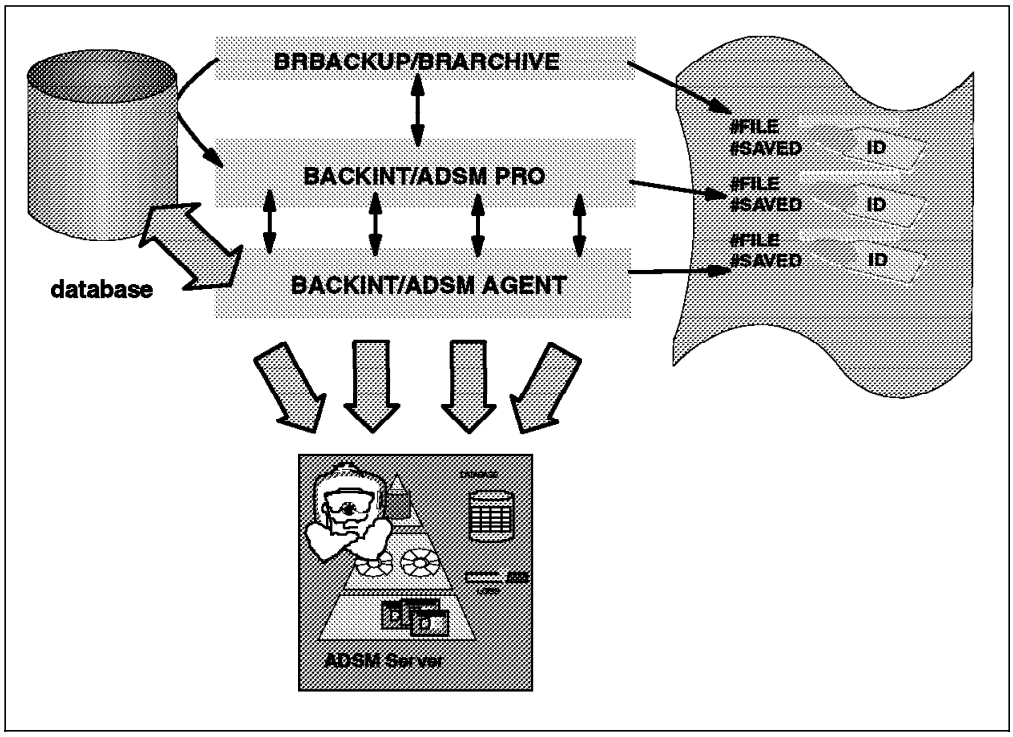


Figure 125. BACKINT/ADSM Architecture

BACKINT/ADSM Version 2 is restructured to use the ADSM API and is split into two modules, BACKINT/ADSM PRO and BACKINT/ADSM AGENT, which communicate with each other through an internal protocol.

- BACKINT/ADSM PRO

  BACKINT/ADSM PRO interfaces to SAPDBA and analyzes the SAP and BACKINT profiles to set up the proper environment for backup and restore to ADSM. It controls one or more BACKINT/ADSM AGENTs by receiving and sending commands and lists of database objects. The database objects can be located on different systems, for example, in an OPS environment on the RS/6000 SP.

- BACKINT/ADSM AGENT

  The BACKINT/ADSM AGENT together with the ADSM API provides the interface to the ADSM server for backup and restore operations. The AGENT reads and writes database objects to and from disk during backup or restore and transfers the data to and from the ADSM API client, which in turn connects to one or more ADSM servers.

A backup and restore solution with SAP R/3, Oracle, BACKINT/ADSM, and ADSM servers consists of several components that interact during a backup or recovery scenario. Figure 126 on page 267 shows the components involved in SAP R/3 backup and restore operations.

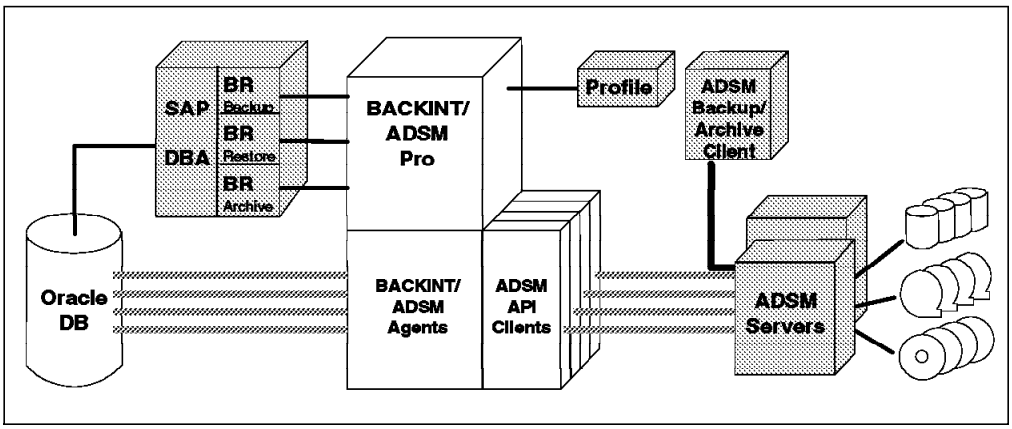


Figure 126. SAP R/3 and BACKINT/ADSM Components

SAPDBA is the primary interface through which SAP DBAs perform database operations in a SAP R/3 environment. For backup, restore, and recovery operations, SAPDBA connects to the Oracle database and, using the SAP R/3 commands BRARCHIVE, BRBACKUP, and BRRESTORE, performs operations.

The SAP R/3 commands read the SAP R/3 profile *initSID.sap* to determine whether the data is to be sent to a local tape or to an ADSM server. If the backup device type is ADSM, SAPDBA passes the list of files to BACKINT/ADSM. BACKINT/ADSM PRO reads its profile, *initSID.utl*, and analyzes the parameters defining the backup environment. BACKINT/ADSM PRO creates one or more BACKINT/ADSM AGENT sessions that interface to the ADSM API. The BACKINT/ADSM AGENT sends the data to one or more ADSM servers based on what is specified in the SAP R/3, BACKINT/ADSM, and ADSM profiles.

10.1.3.1 BACKINT/ADSM Features

The significant features of BACKINT/ADSM Version 2.2. are:

- Database backup version control

  This feature lets you keep a specified number of backups. Several customers have requested the capability to keep SAP R/3 backups by number of versions instead of by expiration date. This function assigns a new version number every time a request from BRBACKUP completes successfully. The version number is assigned to all subsequent redo log backups from BRARCHIVE until the next successful call from BRBACKUP increments the version number again. The backup version number assigned to a partial backup is the same number that was assigned to the last successful full backup.

  A maximum version count can be specified. When this count is reached, BACKINT deletes excess versions from the ADSM server.

- Password handling

  To protect the backup copies from unauthorized access, ADSM provides password authentication for backup and restore operations. BACKINT/ADSM Version 2 stores the current password, encrypted, in its configuration file.

- Individual tablespace locking during online backup

  To use the option to have BACKINT/ADSM lock individual tablespaces during online backups, you need to have the *util_file_online* SAPDBA function enabled. The recovery time from an online backup is shortened because the number of redo logs generated during the online backup process is

significantly reduced. When you use BACKINT/ADSM with locking turned on, the start and end of backup for each file are reported back to SAPDBA. The table space locking allows table spaces to be locked only while the file is being processed for backup. After the backup completes for the file, the table space is unlocked, which reduces the output to the archive logs.

You can think of this function as "just-in-time alter tablespace begin backup." The table space that is to be backed up is set to backup mode just before the backup starts. All of the other table spaces are in normal operating mode. When the backup of the table space is complete, the table space is reset to normal mode, and the next table space is set to backup mode.

- Support for raw devices

  Oracle databases installed on raw devices are supported either whole or partially mixed with file systems.

- Support for virtual shared disks on RS/6000 SP

  SAP R/3 using OPS installed on RS/6000 SP is now supported. Backup is performed through the administrative node of the parallel Oracle configuration. The backup processes are split across VSD nodes.

- BACKINT/ADSM GUI

  A new BACKINT/ADSM user interface called *backfm* helps you query the ADSM server about previously performed backup operations and SAP R/3 files stored at one or more ADSM servers. This utility simplifies the inquiry, restore, and deletion process with BACKINT/ADSM by internally calling the standard BACKINT/ADSM executables to perform all operations. It consists of a text mode window, which displays all backup IDs found on the ADSM servers and all files belonging to a particular backup ID. The backup IDs and files can be easily selected for restore and delete processes.

- Querying the ADSM Server with BACKINT/ADSM Version 2

  Now that BACKINT/ADSM Version 2 uses the ADSM API, you cannot use the ADSM backup-archive client to check which data is stored on the ADSM server. BACKINT/ADSM Version 2 provides a new function to query the ADSM server. You can inquire about all backup IDs, all files for a particular backup ID, and a list of all backup IDs for a specific file, and you can verify whether a specific file is saved under a specific backup ID.

For a complete description of all BACKINT/ADSM features, refer to the *BACKINT/ADSTAR Distributed Storage Manager Program Descriptions and Operation* manual.

10.1.3.2 BACKINT/ADSM Performance Enhancements

BACKINT/ADSM Version 2.2 has introduced further performance enhancements:

- Parallel backup and restore

  BACKINT/ADSM provides a means for maximum data throughput to maximize the speed of recovery and shorten the backup window to reduce the impact on normal system operation. Furthermore, BACKINT/ADSM exploits multiple storage devices, if available, for reading and writing data. BACKINT/ADSM can establish one session per storage device and use the sessions in parallel. BACKINT/ADSM balances the data streams through these sessions.

  The SAP R/3 BRBACKUP, BRARCHIVE, and BRRESTORE commands compile a temporary list of objects. BACKINT/ADSM reads the temporary file list and

splits up the objects into multiple data streams. Then BACKINT/ADSM initiates multiple sessions to ADSM and sends the data through multiple sessions to the ADSM server. If the Oracle data files are stored on different volumes (disks), BACKINT/ADSM optimizes the data transfer by reading the data files from disk in parallel and backing them up in parallel sessions to ADSM.

- Alternative backup paths

  To improve availability of the backup and restore process and reduce network-induced bottlenecks for backup and restore performance, BACKINT/ADSM uses multiple communication paths for data transfer with the ADSM server. These paths can be used simultaneously to increase data throughput, or alternatively, when one or several paths are down.

  Multiple paths are specified in the BACKINT/ADSM profile by path name and number of parallel sessions per path. The number of parallel sessions per path enables BACKINT/ADSM to adjust for different network speeds or to distribute the load over different parts in the network.

  Note: Parallel sessions per path is a subdivision of the total number of sessions determined by the number of storage units (tape or disk drives) available for the backup or restore.

  Using the first path in the profile, BACKINT/ADSM attempts to establish communication with the ADSM server. If successful, it starts the number of parallel sessions specified for this path. The path is skipped if it is inactive. BACKINT/ADSM then repeats this process for the next path until as many sessions as specified by the total session number are activated. Remaining paths are not used. If, because of several inactive paths, the number of sessions that could be started is less than the specified number of total sessions, BACKINT/ADSM terminates the backup job because the job cannot be completed within the backup window.

- Alternative backup servers

  For performance, disaster recovery, and availability purposes, you can use an alternative ADSM server to store your SAP R/3 files. This function is very similar to alternative backup paths, but BACKINT/ADSM sends the data to physically different ADSM servers. BACKINT/ADSM keeps track of all backups and knows on which ADSM server they have been stored. To improve data transfer performance, BACKINT/ADSM accesses each server, using an alternative backup path.

- Multithreaded code

  In Version 2.2 BACKINT/ADSM has been redesigned to operate in multithreaded mode and to be multiprocessor efficient. On symmetric multiprocessor systems, this programming model has significant performance advantages as it uses the hardware and software such as CPU, disk I/O, and network resources more efficiently.

- Null-block compression

  BACKINT/ADSM performs null-block compression before sending the data to ADSM. This lightweight compression method is well suited for database files, sometimes called sparse files, because they usually contain large numbers of null blocks with no data in them. Null-block compression compresses the empty blocks, sending only those blocks with real data within them to the ADSM server.

- With the multiplexing feature, BACKINT/ADSM reads from several files in parallel and stores the files in a multiplexed file on the ADSM server. This multiplexed file consists of several single data files.

10.2 Backup and Restore Concepts

This section describes the different types of backup and restore that can be performed with BACKINT/ADSM.

10.2.1 Online Backup

In an online backup operation, the database is online and in use while a backup of the database, tablespaces, or data files is performed. Before an online backup operation, the BRBACKUP command issues several Oracle and SAP R/3 internal commands to log specific information in the backup log file. For an online backup, Oracle and SAP commands perform the following tasks:

- Copy the primary copy of the control file to the backup copy /oracle/<SID>/sapbackup/<controlfile name>
- Show the environment used for the backup operation
- Show the archive log list before backup
- Show all data files
- Show all online redo log files
- Show all control files
- Show the nondatabase backup parameter utility file: /oracle/<SID>/dbs/init<SID>.utl
- Switch the database tablespaces to backup mode (*ALTER TABLESPACE BEGIN BACKUP*)
- Call the backup interface utility, BACKINT/ADSM

When the online backup operation is finished, the BRBACKUP command issues Oracle and SAP R/3 internal commands to:

- Display all objects that have been sent to one or more ADSM servers
- Switch the database tablespaces to backup-end mode (*ALTER TABLESPACE END BACKUP*)
- Switch the current online redo log
- Show the archive log list after backup
- Call the backup interface utility BACKINT/ADSM again
- Send all nondatabase parameter files required for restore and recovery to ADSM
- Display all parameter files that have been sent to ADSM
- Terminate the backup operation

For an online backup, BRBACKUP creates a read-consistent version of the control file in /oracle/<SID>/sapbackup and saves this file instead of using the open copies of the control files that are in use.

BRBACKUP tracks each backup operation in backup log files stored in the */oracle/<SID>/sapbackup* directory. BACKINT/ADSM uses the backup log files

to send and retrieve the control files, online redo log files, and data files to and from the ADSM server. BRBACKUP also maintains a backup log history file, */oracle/<SID>/sapbackup/backSID.log*, which shows which backup log file to use at restore time and which database objects to restore.

10.2.2 Offline Backup

An offline backup is performed while the database is shut down. Before an offline backup operation, the BRBACKUP command issues several Oracle and SAP R/3 internal commands to log specific information in the backup log file. For an offline backup, Oracle and SAP commands perform the following tasks:

- Perform a startup of the database
- Show the environment used for the backup operation
- Show the archive log list before backup
- Show all data files
- Show all online redo log files
- Show all control files
- Show the nondatabase backup parameter utility file called */oracle/<SID>/dbs/init<SID>.utl*
- Perform a shutdown of the database
- Call the backup interface utility, BACKINT/ADSM

When the offline backup operation is finished, the BRBACKUP command issues Oracle and SAP R/3 internal commands to:

- Display all objects that have been sent to one or more ADSM servers
- Perform a startup of the database
- Show the archive log list after backup
- Call the backup interface utility BACKINT/ADSM again
- Send all nondatabase parameter files required for restore and recovery to ADSM
- Display all parameter files that have been sent to ADSM
- Terminate the backup operation

At a full offline backup BRBACKUP also includes the online redo log files because no recovery is necessary after a full offline backup. The backup is consistent.

Offline Backup Tip

Stopping and restarting SAP have a significant impact on its performance. SAP uses a large amount of real and virtual memory to buffer data between itself and its database. The data in the buffers is built up over time. If you restart SAP, the data in the buffers is lost, and it takes time to repopulate the buffers and regain your pre-shutdown performance level. SAP V3.0E has a feature that enables you to stop and restart Oracle without affecting SAP and its buffers. However, this feature was not tested.

The way the buffer clearing works affects how often you would want to run offline backups because your system performance will be affected after each offline backup. We recommend that you run online backups as much as possible.

10.2.3 Full Backup

A full backup is a backup of all database objects of the SAP R/3 Oracle database including the following files:

- The read-consistent version of the control file
- All data files
- Online redo log files
- All parameter files

A full backup can be taken in online or offline mode.

10.2.4 Partial Backup

A partial backup is a backup of one or more data files or tablespaces within the database. A partial backup can include the following:

- A read-consistent version of the control file
- Data files
- Tablespaces
- Parameter files

A partial backup can be taken in online or offline mode. A partial backup is based on Oracle database objects where you can specify which data files or tablespaces you want to back up. When backing up a data file or tablespace, BRBACKUP also adds the required control file and parameter files to the backup object list.

10.2.5 Archive Redo Log Backup

The archive redo logs, also called *offline redo logs*, are critical for recovery of the database and database objects. Oracle produces archive redo logs by copying the contents of an online redo log to an external file after the online redo log file has filled up. Each archive redo log is given a unique name by Oracle. Protection of these archived logs is critical. The disk volumes on which they are stored should be mirrored for maximum availability. It is also very important to back up the archive redo logs to ADSM as soon as possible after they have been created.

Backup of the archive redo logs is performed through the BRARCHIVE command, which can save the archive redo logs to tape, disk, or ADSM. To free up disk space, the archive redo logs may be deleted after they are backed up.

BRARCHIVE is the recommended method of backing up the archived redo logs as it allows them to be backed up and restored with the SAPDBA utility. An alternative and supplementary method is to schedule regular incremental backups of archived redo logs by using the ADSM backup-archive client. This is not a substitute for using BRARCHIVE because SAPDBA cannot be used to restore these backups. However, it is a valid method of taking additional backups of the archived redo logs for disaster recovery purposes.

The archive redo logs are often used to implement an incremental backup strategy. Between two full database backups, a backup of the archive redo logs is often referred to as an incremental backup. However, this is not a real incremental backup solution, but is often implemented, especially if the database is very large and a daily full backup is not possible.

10.2.6 Database Recovery

SAPDBA performs database recovery of a SAP R/3 Oracle database, using the same functions as Oracle uses natively. SAPDBA provides the SAP R/3 administrator with several recovery menus and invokes the Oracle internal recovery commands. SAPDBA performs several Oracle commands to check whether all database objects are present and working properly. If an object is missing, corrupt, or not in an appropriate state, SAPDBA suggests the right recovery activities for the database administrator to perform to fix the problem.

As BRBACKUP just performs a backup of the primary file copies, SAPDBA creates mirror copies of the control files and online redo log files. After all primary copies have been restored properly, SAPDBA copies the specific files to their mirror destination directories.

SAPDBA logs all restore operations in a restore log file stored in the /oracle/<SID>/sapbackup directory and the recovery operations in a recovery log file stored in the /oracle/<SID>/sapreorg directory.

10.2.6.1 Restore without Recovery

You can perform a restore without recovery of the database if the database backup taken is an offline backup. Because an offline backup always represents a consistent version of the database, no additional recovery steps are needed. SAPDBA does not restore archive redo logs by default during a restore without recovery operation.

For a full database restore without recovery, the online redo logs and the control files will be restored because they are consistent and represent a consistent view of the database. Because of this no roll-forward recovery is required.

10.2.6.2 Restore with Recovery

You can perform a restore with recovery of the database if the database backup is:

- A full online backup
- A partial backup of individual tablespaces or data files

An online backup always creates an inconsistent backup of the database. Thus you must perform recovery after restoring all data files and control files. You can perform the following recovery options:

- Until a specific point in time (YYYY-MM-DD)
- Until now (most current version)

- Recovery using the Oracle BACKUP CONTROLFILE option

For a restore with recovery, BRRESTORE will not restore the online redo logs because the information in the online redo logs is not consistent and will be overwritten when applying the archive redo logs.

10.2.7 Nondatabase Files

All files that do not belong to Oracle internal database objects such as data files, redo log files, or control files are called nondatabase files. Nondatabase files can be:

- SAP R/3 parameter files and executables
- Oracle parameter files and executables
- BACKINT/ADSM parameter files and executables
- Log files of an archive, backup, restore, or recovery process

When the BRBACKUP command backs up data to BACKINT/ADSM, it also saves some SAP R/3 and Oracle-specific nondatabase files:

- initSID.dba
- initSID.utl
- initSID.sap
- initSID.ora

In addition to these initialization files, SAP creates log files for previously run backup and archive operations. These are also backed up from the following directories:

- /oracle/<SID>/sapbackup
- /oracle/<SID>/saparchive
- /oracle/<SID>/sapreorg

The SAP R/3 executables are stored in /usr/sap and /sapmnt. The Oracle executables are stored in /oracle/<SID>. These are not backed up by BACKINT/ADSM.

10.2.8 BRBACKUP Process

The BRBACKUP command is a SAP R/3 utility to back up Oracle database objects and nondatabase files. With BRBACKUP you perform full or partial backups of the Oracle database in online or offline mode.

BRBACKUP runs from either SAPDBA or the command line. Figure 127 on page 275 shows the components involved with a SAP R/3 backup operation performed by BRBACKUP.

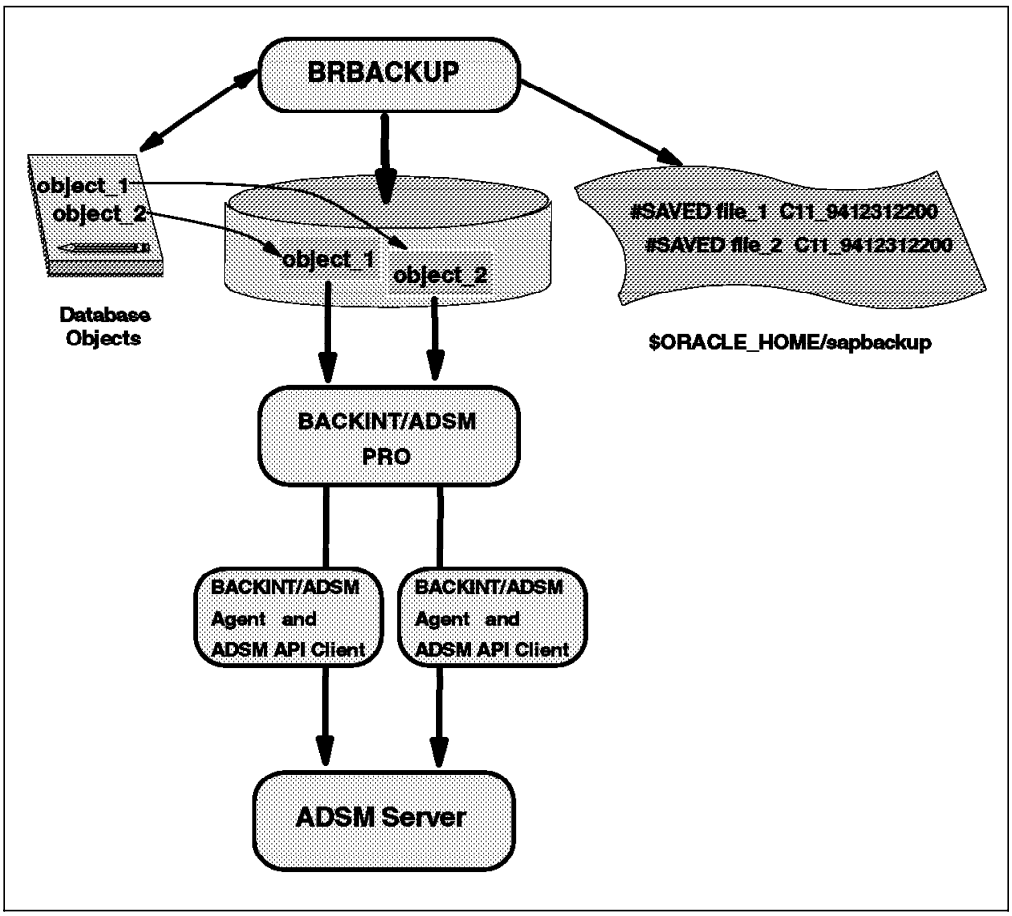


Figure 127. SAP R/3 BRBACKUP Process

BRBACKUP compiles a list of objects to save and passes this list to BACKINT/ADSM PRO. BACKINT/ADSM PRO sends the object list to the BACKINT/ADSM AGENT and creates a protocol file in $ORACLE_HOME/sapbackup. The BACKINT/ADSM AGENT then reads the database objects and sends them to the ADSM server.

At the end of a backup operation, BRBACKUP sends the created backup log file to ADSM, using BACKINT/ADSM.

10.2.9 BRRESTORE Process

The BRRESTORE command is a SAP R/3 database utility that manages the restore and recovery process.

BRRESTORE is invoked by SAPDBA and connects to BACKINT/ADSM in the same way as BRBACKUP and BRARCHIVE. SAP recommends not using the BRRESTORE command on the command line, but from SAPDBA, because SAPDBA includes the necessary recovery steps to recover a restored object. BRRESTORE does not include the recovery operation, however.

When you restore a database, you might perform the restore operation using SAPDBA menus. SAPDBA invokes the BRRESTORE command to restore the:

- Required parameter and backup log files
- Data files

- Control files
- Required archive redo logs

Figure 128 shows the components involved with a SAP R/3 restore operation.

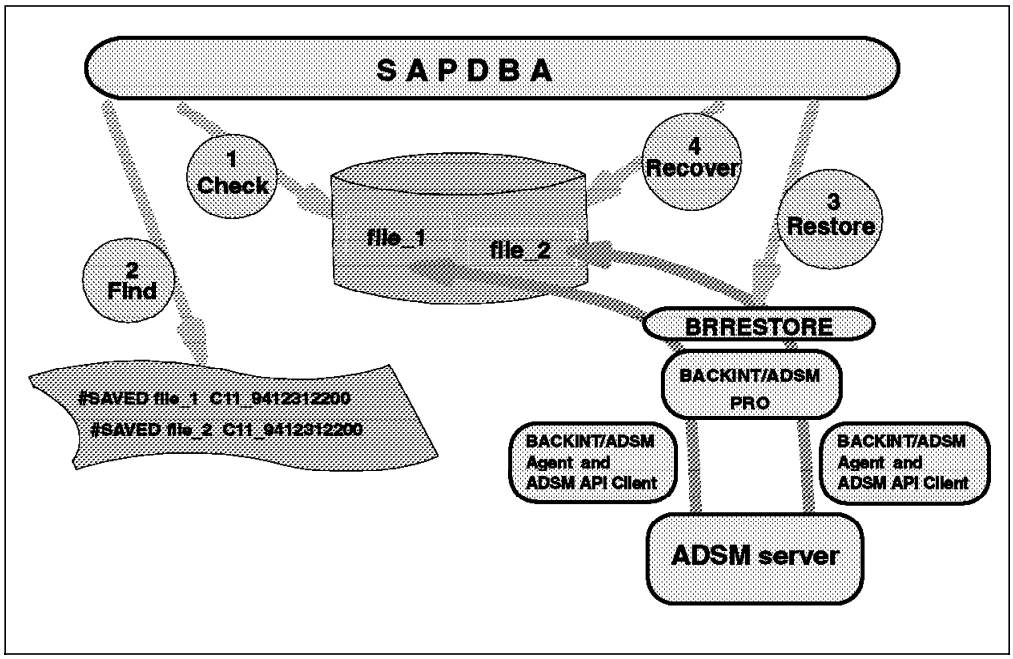


Figure 128. SAP R/3 BRRESTORE Process

1 SAPDBA invokes Oracle internal commands to check the control files, online redo logs files, and data files of the SAP R/3 Oracle database.

2 From the appropriate backup log file, SAPDBA identifies the files to restore.

3 SAPDBA invokes BRRESTORE and passes the list of files to restore to BACKINT/ADSM PRO, which calls the BACKINT/ADSM AGENT to restore the files from ADSM through the ADSM API.

4 After control is passed back to SAPDBA, SAPDBA executes Oracle internal commands to recover the database.

10.2.10 BRARCHIVE Process

The BRARCHIVE command is a SAP R/3 database utility that manages the Oracle archived redo logs. It connects to BACKINT/ADSM in the same way as BRBACKUP connects to it.

BRARCHIVE saves and deletes archived redo logs from the filesystem to the target backup device and maintains an archive log history in a file: /oracle/SID/saparch/archSID.log. In conjunction with BACKINT/ADSM, the archive redo logs can be archived to two separate storage pools on the ADSM server by using two management classes to define the storage pools to be used. This is done to minimize the risk of losing archive redo logs. The archive log history contains the archived redo log numbers that have been copied or moved from the Oracle database archived redo log destination directory to the backup destination. BRARCHIVE uses an internal identification string to determine which archived redo logs to restore to disk.

BRARCHIVE also archives the necessary SAP R/3 and Oracle parameter files needed for restore and recovery.

10.3 SAP R/3 and BACKINT/ADSM Tools

In this section we describe the tools that enable you to perform backup and recovery operations of SAP R/3 Oracle databases. First we cover the main functions of the commands used to initiate backup and recovery tasks. We then introduce the most common profile options that control the behavior of these tools.

Please note that the syntax diagrams presented in this section reflect our experience of a valid combination of options and values when using the SAP R/3 utilities with BACKINT/ADSM. When reviewing the printed SAP R/3 documentation or the online help facilities, you will find a greater number of options that we did not use.

The following commands and profiles are discussed in this section:

BRBACKUP Used to back up the database files, such as data files, online redo log files, and the control file

BRARCHIVE Used to store archived (offline) redo logs to ADSM and optionally delete them from the archive directory

BRRESTORE Used to perform restore operations for archived redo logs, data files, tablespaces, or a database

SAPDBA Provides an ASCII menu for BRBACKUP, BRARCHIVE, and BRRESTORE, recovery capabilities, and additional Oracle database administration functions

stopsap Used to stop SAP R/3 dialog instances and the database server

startsap Used to start SAP R/3 dialog instances and the database server

backfm Provides an ASCII menu for individual file restore.

init*SID***.sap** Profile for BRBACKUP, BRARCHIVE, BRRESTORE, and SAPDBA

init*SID***.utl** Profile for BACKINT/ADSM and backfm

10.3.1 User Accounts and Groups

The central instance of a SAP R/3 system typically has two user accounts, the *SID*adm user and the ora*SID* user.

The *SID*adm user is the SAP R/3 dialog instance administrator. In our environment the user name is C21adm. You use this user to stop and start SAP R/3 dialog instances and the Oracle server instance. The *SID*adm user belongs to the sapsys and dba groups. These are the commands you use with this user account:

- startsap
- stopsap

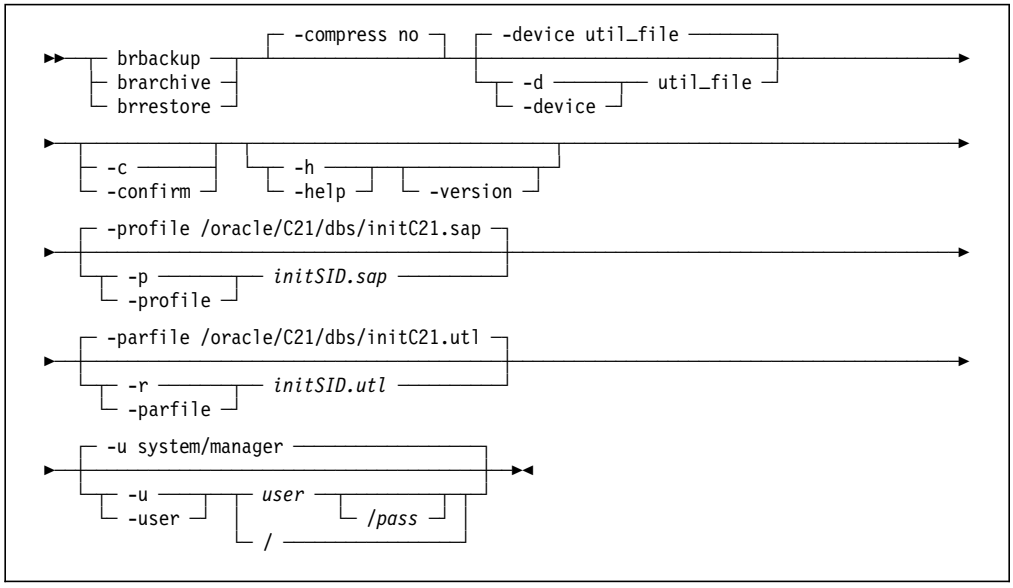
The ora*SID* user is the Oracle database administrator. The user name in our environment is orac21. You use this user account to perform all backup and recovery operations. Likewise, this user is also used to start and stop the Oracle database instance. The commands you use under the authority of ora*SID* are:

- BRBACKUP
- BRARCHIVE
- BRRESTORE
- SAPDBA
- backfm

In a default SAP R/3 installation, the Oracle administrator belongs to the dba group, and the SAP R/3 tools belong to the sapsys group. Nevertheless, the ora*sid* administrator can use the SAP R/3 tools because the default installation leaves the SAP R/3 tools with the file mode set to execute permission for all users.

10.3.2 Common Options

The options listed in this section are common to BRBACKUP, BRARCHIVE, and BRRESTORE. The default options (indicated as a path above the main line in the syntax diagrams) are based on our modified initC21.sap profile. You can change the default settings by editing the profile as described later in this section.



-compress no

This option controls whether the SAP R/3 utilities use the UNIX compress command to reduce the amount of data to process. You cannot use this option when using BACKINT/ADSM. Instead, use the RL_COMPRESSION option described later in this section.

-device util_file

This option specifies that the data storage operation is performed by BACKINT/ADSM. The SAP R/3 utilities pass all backup and restore requests to the ADSM server, using the BACKINT/ADSM interface.

-confirm

This option suppresses messages where the SAP R/3 tools expect an interactive user confirmation. You use this option to enable the unattended mode of the SAP R/3 tools, BRBACKUP and BRARCHIVE, for example, when issuing the BRBACKUP command by means of a scheduled UNIX shell script. The BRRESTORE command cannot run totally unattended.

Even if you use this option, BRRESTORE still requires user input to continue certain operations.

-help

This option displays an overview of all available options for the corresponding command. Many of the options you see are obsolete, when using the SAP R/3 tools in conjunction with BACKINT/ADSM. The additional option, *version*, displays the software level of the SAP R/3 tool.

-profile init*SID*.sap

With this option you can specify a profile other than the default profile for the SAP R/3 tools. This option is useful if you want to keep the ADSM-related settings in a separate location. For example, in an environment where the SAP R/3 tools are used to perform local disk operations, you can define a separate init*SID*.sap.adsm, which has the default definitions needed to perform backups to ADSM storage.

-parfile init*SID*.utl

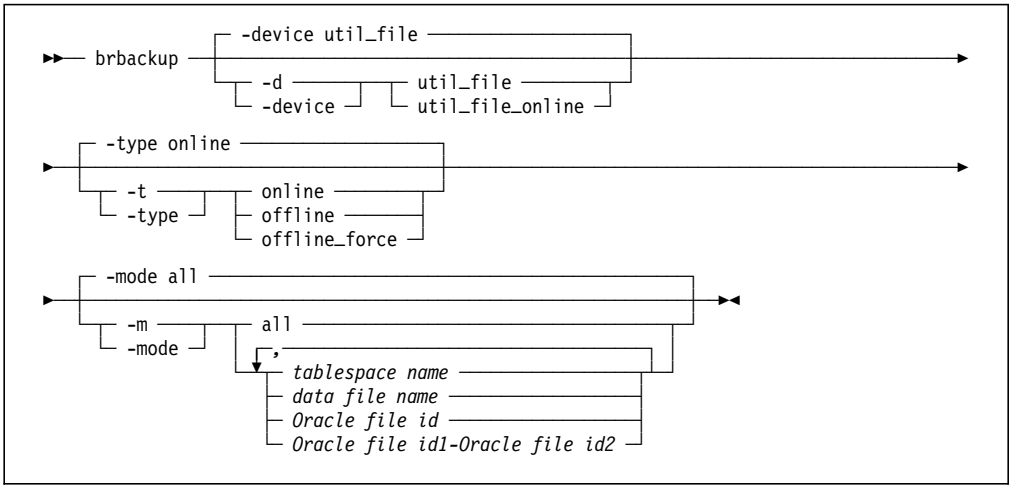
This option enables you to use a specific BACKINT/ADSM profile, other than the default profile.

-user/password

The SAP R/3 BRBACKUP, BRARCHIVE, BRRESTORE, and SAPDBA commands connect to the Oracle database instance when performing operations. This is why you need an Oracle user name and a password in addition to the operating system authority required to execute the command. The SAP R/3 installation defaults are system for the user name and manager for the password. If not otherwise specified, the above commands use these values.

10.3.3 BRBACKUP

The BRBACKUP command performs backup functions for database files. Based on the chosen operation, the backup includes data files of the database, the control file, and online redo log files. Every BRBACKUP operation is logged into an individual log file called the detail log, which resides in the /oracle/*SID*/sapbackup directory. Also, a summary line containing the name of the detail log file, the start and end time, and the BRBACKUP return code is appended to the /oracle/*SID*/sapbackup/back*SID*.log file. This syntax diagram shows common options used when performing backups with BACKINT/ADSM:



-device

The *util_file* option specifies that the backup operation is performed by BACKINT/ADSM under the control of BRBACKUP. BRBACKUP performs all database-relevant tasks, such as altering the backup mode of tablespaces during online backups, while the actual data transfer is performed directly to the ADSM server, using BACKINT/ADSM. The alternative option, *util_file_online*, activates the just-in-time altering of tablespaces during an online backup. Therefore only those tablespaces currently being backed up are altered to the backup mode, rather than changing all tablespaces at the beginning and end of an online backup.

-type

With this option, you can control whether BRBACKUP performs an online or offline backup. When performing an online backup, BRBACKUP alters the backup mode of tablespaces according to the device option (util_file or util_file_online). When you select an offline backup, BRBACKUP shuts down the Oracle instance before initiating the backup operation. You have to stop dialog instances before performing an offline backup; otherwise the backup will fail. When using the *offline_force* option, BRBACKUP does not check for active SAP R/3 system users.

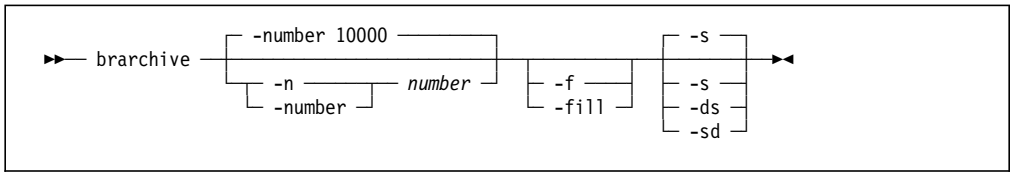
-mode

This option controls which objects BRBACKUP includes in the backup job. The default value backs up all files of the database. That is, during an online backup, the data files and control files are backed up. Additionally, during an offline backup the online redo logs are backed up. Optionally, you can perform a partial backup by specifying which objects you want to include in the backup.

10.3.4 BRARCHIVE

The BRARCHIVE command is used to manage archived redo logs. Archived redo logs are redo logs that have been written to the /oracle/*SID*/saparch archive directory and are no longer in use by the Oracle instance. Using BRARCHIVE, you can store these redo logs within the ADSM server storage and delete the original log files from your archive directory. Every BRARCHIVE operation is logged in a detail log file also stored in the /oracle/*SID*/saparch/ directory. Additionally, the /oracle/*SID*/saparch/arch*SID*.log file contains historical information about every redo log. This information includes the time the redo log has been archived by the Oracle instance, whether it has been stored to ADSM storage, and the time the archive log has been deleted from the archive directory.

This syntax diagram shows the options used with BRARCHIVE in conjunction with BACKINT/ADSM:



-number *number*

With this option you specify the *number* of redo logs to process with the BRARCHIVE run. The default value is 10000, which basically means that all redo logs are processed.

-fill

When you use the fill option, BRARCHIVE waits for additional redo logs to be archived into the archive directory and processes them immediately, until the total count of redo logs specified by the -number option is reached.

-fill stop

You can use this option to regularly stop a *brarchive -fill* started in the background (instead of just killing the process).

-s (save archive logs)

This option writes archived redo logs to ADSM storage.

-ds (delete saved archive logs)

This option deletes from your local archive directory archived redo logs that have been successfully written to ADSM during a previous BRBACKUP run.

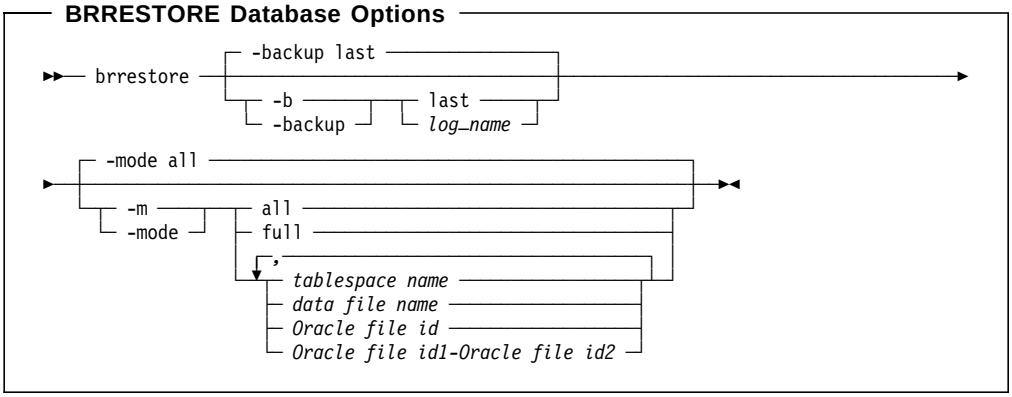
-sd (save and delete archive logs)

This option writes archived redo logs to ADSM and deletes the local copy from your archive directory immediately after each redo log file has been stored successfully within ADSM server storage.

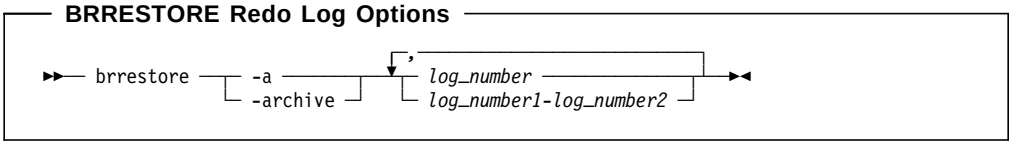
10.3.5 BRRESTORE

The BRRESTORE command restores a database or individual database files, using backups previously performed with BRBACKUP. Also, you can use the BRRESTORE command to retrieve specific archive logs from ADSM storage. Every operation performed with BRRESTORE generates a log file stored in the /oracle/*SID*/sapbackup directory.

The following syntax diagrams give an overview of the options you can use with BRRESTORE and BACKINT/ADSM:



or



-backup

With this option you specify the name of a BRBACKUP detail log file. BRRESTORE then restores the requested objects, using this specific

backup. Only detail logs of successfully terminated BRBACKUP runs are eligible for this option. If this option is omitted, BRRESTORE restores the most recent backup.

-mode

When using this option, you define which objects the BRRESTORE command restores from ADSM storage. The default value *all* restores all data files, but not the control files and online redo log files. The *full* value restores all database files including the online redo logs and the control files. This mode also re-creates the mirror copies of the control files. Alternatively, BRRESTORE enables you to specify individual database files for restore.

-archive

To retrieve individual redo log files from ADSM, you can use the *archive* option. You can specify specific redo logs and a range of log files.

10.3.6 SAPDBA

The SAPDBA tool is used to perform Oracle database administrative tasks. It provides ASCII menus for the BRBACKUP, BRARCHIVE, and BRRESTORE commands, additional recovery capabilities, and other administrative functions. Figure 129 shows the main menu of the SAPDBA tool.

```
SAPDBA V3.0F - SAP Database Administration

                ORACLE version : 7.2.3.0.0
                ORACLE_SID     : C21
                ORACLE_HOME    : /oracle/C21
                DATABASE       : Database open
                SAPR3          : not connected

      a - Startup database             h - Backup database
      b - Shutdown database            i - Backup archive logs
      c - Tablespace extension         j - Check (and repair) database
      d - Reorganization               k - Restore/Recovery
      e - Export/import                l - Show/Cleanup
      f - Archive mode                 m - Expert mode
      g - Additional functions         n - Help

      q - Quit

      Please select ==>
```

Figure 129. SAPDBA Main Menu

The menu structure you use when setting up and performing backup and recovery operations is outlined in Figure 130 on page 283. Note that this tree view shows only those menus used frequently when you perform backup and restore operations.

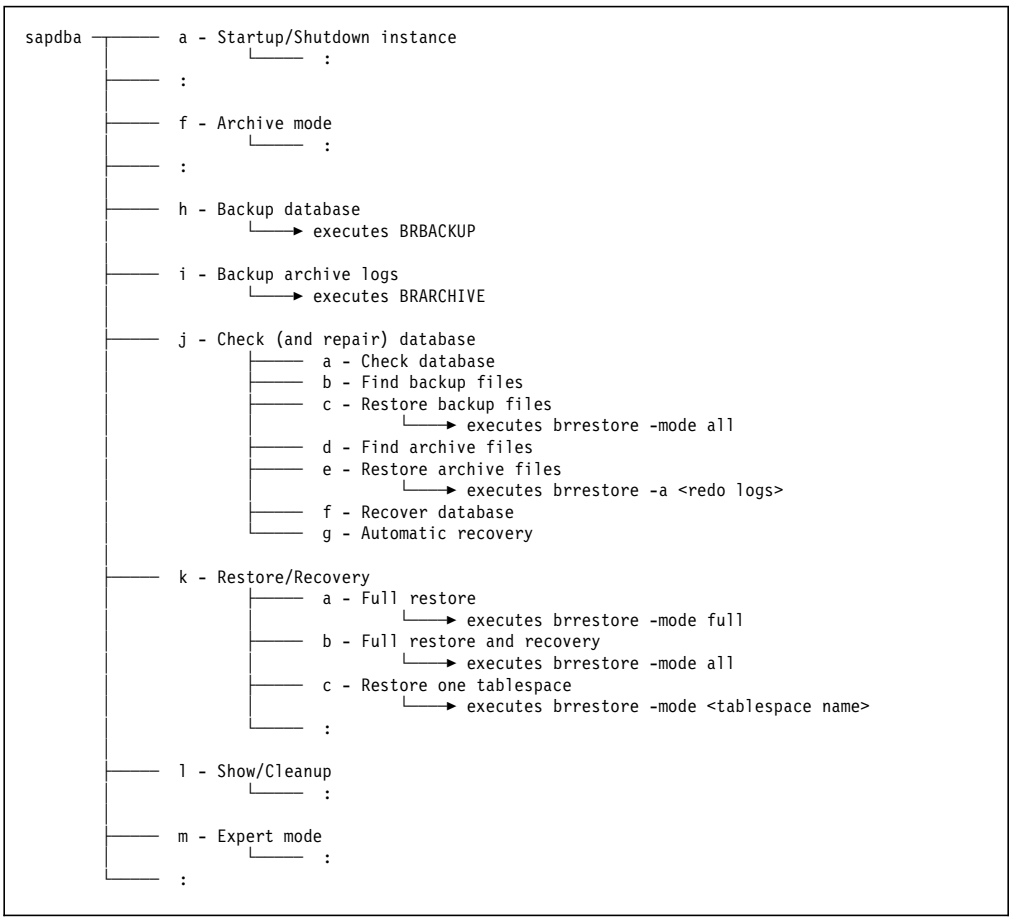


Figure 130. SAPDBA Menu Structure

Startup/Shutdown instance
Used to start and stop the Oracle database instance

Archive mode
Used to toggle the Oracle log mode (archive mode or noarchive mode). To perform online backups, you have to enable the archive mode.

Backup database
ASCII front end for the BRBACKUP command

Backup archive logs
ASCII front end for the BRARCHIVE command

Check (and repair) database
This menu is used to identify damaged database files (a) and restore them from ADSM storage (b+c). In addition, the database is recovered up to the current time (f), using the required archive redo logs (d+e). You can trigger each step manually, or you can perform an automatic recovery, which prompts you through the required steps (a-f).

Restore/Recovery
This menu provides the functions you need if you want to restore an older database backup or perform point-in-time recovery.

Show/Cleanup
Used to view or delete BRBACKUP and BRARCHIVE profiles and logfiles

Expert mode

This menu enables you to protect certain SAPDBA functions with a password.

10.3.7 BACKINT Filemanager

The BACKINT filemanager, backfm, enables you to perform BACKINT/ADSM operations without the SAP R/3 utilities. The backfm functions include viewing, restoring, and deleting individual files previously backed up with BRBACKUP or BRARCHIVE. Figure 131 shows an example of a backfm screen.

```
                  BACKINT-Filemanager V1.3, Copyright IBM 1998
 .------------------+-------------------------------------------------------.
 |   Backup-ID's    | Files stored under SAP___9802251521                   |
 |------------------+-------------------------------------------------------|
 | SAP___9802251604 |  /oracle/C21/sapbackup/cntrl.dbf                      |
 | SAP___9802251521 |  /oracle/C21/sapdata1/ddicd_1/ddicd.data1             |
 | SAP___9802251435 |  /oracle/C21/sapdata1/pooli_1/pooli.data1             |
 | SAP___9802251416 |  /oracle/C21/sapdata1/roll_1/roll.data1               |
 | SAP___9802251327 |  /oracle/C21/sapdata1/system_1/system.data1           |
 | SAP___9802250959 |  /oracle/C21/sapdata1/temp_1/temp.data1               |
 | SAP___9802250958 |  /oracle/C21/sapdata2/clud_1/clud.data1               |
 |                  |  /oracle/C21/sapdata2/es30fi_1/es30fi.data1           |
 |                  |  /oracle/C21/sapdata2/sourced_1/sourced.data1         |
 |                  |  /oracle/C21/sapdata2/user1i_1/user1i.data1           |
 |                  |  /oracle/C21/sapdata3/btabi_1/btabi.data1             |
 |                  |  /oracle/C21/sapdata3/clui_1/clui.data1               |
 |                  |  /oracle/C21/sapdata3/docui_1/docui.data1             |
 |                  |  /oracle/C21/sapdata3/loadi_1/loadi.data1             |
 |                  |  /oracle/C21/sapdata3/proti_1/proti.data1             |
 |                  |  /oracle/C21/sapdata3/stabd_1/stabd.data1             |
 |------------------+-------------------------------------------------------|
 |  7 BIDs          | 28 File(s) - 0 marked                                 |
  ------------------+-------------------------------------------------------'
 TAB change windows  F2 Restore    F5 reFresh      F8 Delete           F10 eXit
 UP,DOWN scroll      F3 Mark all   F4 Unmark all   ENTER mark file
```

Figure 131. backfm Sample Screen

10.3.8 init*SID*.sap Profile Options

The init*SID*.sap profile is used to define default settings for the BRBACKUP, BRARCHIVE, and BRRESTORE utilities. The settings for the profile in our environment are:

```
# cat /oracle/C21/dbs/initC21.sap
  ⋮
backup_type = online
backup_dev_type = util_file
compress = no
util_par_file = ?/dbs/init@.utl
  ⋮
```

The *backup_type* setting is the same as the BRBACKUP -t or -type option (10.3.3, “BRBACKUP” on page 279). The *backup_dev_type* and *util_par_file* settings are the same as the -d/-device and -r/-parfile common options (10.3.2, “Common Options” on page 278).

10.3.9 init*SID*.utl Profile Options

The profile init*SID*.utl, in our environment, /oracle/C21/dbs/initC21.utl, controls the operation of BACKINT/ADSM. Here is an example of the main options you need to customize to set up an operative BACKINT/ADSM environment:

```
  ⋮
MAX_SESSIONS    1
BACKAGENT       /sapmnt/C21/exe/backagent
REDOLOG_COPIES  2
CONFIG_FILE     /oracle/C21/dbs/initC21.bki
RL_COMPRESSION  YES
MULTIPLEXING    3
MAX_VERSIONS    5
REPORT          2

SERVER teraserv
    SESSIONS            1
    PASSWORDREQUIRED    NO
    BRBACKUPMGTCLASS    r3_db
    BRARCHIVEMGTCLASS   r3_log1 r3_log2
```

MAX_SESSIONS
Specifies the number of tape drives you want to use in parallel, when performing backup and restore operations

BACKAGENT
Specifies the path to the backagent executable

REDOLOG_COPIES
Specifies the number of redo log copies you want to store within the ADSM server. For each redo log copy, you have to specify a corresponding BRARCHIVE management class in this profile.

CONFIG_FILE
This keyword defines the path to the BACKINT/ADSM config file.

RL_COMPRESSION
Using this parameter you can activate the run-length compression performed by BACKINT/ADSM. This reduces the amount of data sent over the network.

MULTIPLEXING
This option controls the level of file multiplexing BACKINT/ADSM performs; that is, the number of I/O threads used to perform disk operations. The total number of I/O threads is calculated as follows:
MULTIPLEXING × *MAX_SESSIONS* = *NUM_IOTHREADS*

MAX_VERSIONS
Specifies the number of backup versions to keep

REPORT
Defines the level of reporting written to the log files

SERVER
Specifies the name of the ADSM server to use. This name refers to the SERVERNAME keyword in the ADSM dsm.sys file.

SESSIONS
Defines the number of sessions to use on a specific server

PASSWORDREQUIRED
With this option you can control whether or not BACKINT/ADSM uses a password to connect to the ADSM server. Turn this option off if you are using the ADSM client option, PASSWORDACCESS=generate.

BRBACKUPMGTCLASS
Specifies the management class to be used on the ADSM server for managing database backup objects created by BRBACKUP.

BRARCHIVEMGTCLASS
Specifies the management classes to use on the ADSM server for managing archive redo logs stored with BRARCHIVE. When using multiple redo log copies (REDOLOGCOPIES), you have to specify a different management class for each redo log copy.

BRBACKUPMGTCLASS

BRBACKUP uses the ADSM archive function when performing backups. The management class defined for BRBACKUPMGTCLASS must have an archive copygroup defined otherwise backups will fail.

10.4 BACKINT/ADSM Installation

SAPDBA Version 3.0F runs on several SAP R/3 releases. The SAP R/3 release is not important for backup and recovery purposes, only the SAPDBA and Oracle database versions are important. In our examples we installed SAP R/3 Release 3.0F with a default loaded Oracle database (approximately 6.3 GB). The following software releases were used:

- AIX Version 4.2.1.0
- ADSM for AIX Version 3.1.0.1 client and 3.1.0.1 server
- SAPDBA Version 3.0F with Oracle Version 7.2.3.0.0
- BACKINT/ADSM Version 2.2.5.4

BACKINT/ADSM is distributed as a tar compressed image that contains:

- The backpro and backagent executables
- The BACKINT/ADSM profile: initSID.utl
- Sample ADSM client profiles: dsm.opt and dsm.sys
- Sample AIX backup shell scripts
- Documentation files

The installation process does not use SMIT. The tar command is used to uncompress the installation image that contains versions of BACKINT/ADSM for the various operating systems. Installation is performed by copying the appropriate version to the target directories. This process is fully described in the *BACKINT/ADSM Program Description and Operations Version 2 Release 2* manual.

10.4.1 SAPDBA Initialization

Before you can use SAPDBA, BRBACKUP, or BRARCHIVE, certain tables have to be initialized in the Oracle database. Initialization is done automatically when you start the SAPDBA program for the first time. First you must ensure that the Oracle database is up and running before SAPDBA is started. You can do this by switching to the SAP admin ID and issuing the startsap db command:

```
> su - c21adm
capeverde:c21adm> startsap db

Starting SAP R/3 C21 Database
-----------------------------
 Startup-Log is written to /home/c21adm/startdb.log
 Database started

Checking SAP R/3 C21 Database
-----------------------------
 Database is running

capeverde:c21adm>
```

Having started the database, you can start the SAPDBA program:

```
capeverde:orac21> sapdba
SAPDBA: The SAP table/index SAPDBAPRIV... for DBA is missing.
Do you allow SAPDBA to create this table/index? (y/n) [n] ==> y
SAPDBA: SAP Table SAPDBAPRIV created.
SAPDBA: 10 of 10 privileges were changed and saved.
Press <return> to continue ...

SAPDBA: SAP Index SAPDBAPRIV0 created.
Press <return> to continue ...

SAPDBA: Structure log file '/oracle/C21/sapreorg/structC21.log' created.

Press <return> to continue ...

SAPDBA: Refreshing the structure log!
SAPDBA: 27 new data file entries written to the structure log
Press <return> to continue ...
```

After you have finished these steps, the SAPDBA main menu appears (Figure 129 on page 282).

10.4.2 Setting the SAPDBA Expert Password

The SAPDBA program enables you to protect certain functions with a password. Menus protected with a password can be used only after you have entered expert mode. Expert mode is enabled by setting the expert mode password. The following SAPDBA menu path sets the password:

```
sapdba ——— m - Expert mode
                └——— b - Set initial password for expert
```

You will see the b option only if the expert password has not already been set. The root user can reset the expert password by deleting the file that stores the expert password:

```
#> rm /oracle/C21/passwd/passwd.dba
```

10.4.3 Oracle Archive Mode Setup

A prerequisite for performing online backups is that the Oracle database must be running in archive log mode. The SAPDBA menu (Figure 129 on page 282) f option, Archive mode, shows you the current mode the database is operating in and enables you to change it:

```
                              Archive Mode

                    LOCAL INSTANCE    : C21
                    DATABASE LOG MODE : archive mode

                    INSTANCE STATE    : open
                    AUTOMATIC ARCHIVAL: enabled
```

The SAP R/3 utilities can now be used.

10.5 SAP R/3 Backup and Recovery Examples

In this section we show examples of using BACKINT/ADSM to back up and restore SAP R/3 Oracle databases.

The SAP R/3 Oracle database consists of several tablespaces (system and user tablespaces), mirrored control files, mirrored online redo logs, and archived redo logs. SAP R/3 Oracle tablespaces are installed in different file systems on separate disks for performance and data security reasons. SAP R/3 generally uses Oracle′s software mirroring facilities to ensure hardware-platform independence. SAP R/3 supports Oracle on file systems, raw devices, and virtual shared disks on RS/6000 SP.

In a default installation you place the data and index tablespaces on separate file systems and physical disks. The three copies of the control files must be distributed over different physical disks, with each control file on a separate disk to prevent the loss of all control files. The online redo logs are also mirrored and distributed over two separate file systems. We give an overview of the file systems and path names where the tablespaces, control files, online redo logs, archived redo logs, and SAP R/3 files are located. The name of the SAP R/3 Oracle database, which Oracle calls *SID*, is set to C21 in our examples. Figure 132 on page 289 illustrates the file and path names defined for our SAP R/3 Oracle environment.

```
/usr/sap/C21           SAP related files and links
/sapmnt/C21            SAP related files, executables
/oracle/C21            Oracle Home directory, Oracle related files, control file

/oracle/C21/origlogA   Online redo log files (first copy)
/oracle/C21/origlogB   Online redo log files (first copy)
/oracle/C21/mirrlogA   Online redo log files (mirror copy)
/oracle/C21/mirrlogB   Online redo log files (mirror copy)

/oracle/C21/sapdata1   tablespaces (system tablespaces, control file)
/oracle/C21/sapdata2   tablespaces (user tablespaces, index tablespaces, control file)
/oracle/C21/sapdata3   tablespaces (user tablespaces, index tablespaces)
/oracle/C21/sapdata4   tablespaces (user tablespaces, index tablespaces)
/oracle/C21/sapdata5   tablespaces (user tablespaces, index tablespaces)
/oracle/C21/sapdata6   tablespaces (user tablespaces, index tablespaces)

/oracle/C21/saparch    Archived redo logs and Archive Log files
/oracle/C21/sapbackup  Backup log files
/oracle/C21/sapreorg   Restore and Recovery log files
```

Figure 132. SAP R/3 File Systems and Directories

The examples we show are:

- Full offline backup of all tablespaces
- Full online backup of all tablespaces
- Backup of archived redo logs
- Managing backups with backfm
- Restore and recover a data file
- Full offline restore of the entire SAP R/3 Oracle database
- Database point-in-time restore

Note: For all SAPDBA, BRBACKUP, and BRARCHIVE backup and restore operations, you must log in as the Oracle database administrator, *orac21*.

10.5.1 Full Offline Backup

A full offline backup is one of the most common ways of backing up an entire SAP R/3 Oracle database. There are several reasons to perform a full offline backup of the entire database including the parameter files:

- To save the current version of the database before a software upgrade (in SAP R/3 you call a software upgrade “put”)
- To ensure a consistent image of the entire database
- To keep a consistent version of the database as the starting point of an incremental backup strategy in which you save only the archived redo logs
- It is one of the easiest and most consistent ways of recovering after a complete loss of all data.

Usually a scheduling facility, such as AIX cron or the ADSM scheduler, starts a full offline backup of the entire database. Therefore, we show an example using BRBACKUP commands.

Note: If you start a job using the AIX cron facility, the command started by AIX cron has root user privileges. You can switch to the orac21 user without being prompted for a password.

Switch to the Oracle orac21 user, using the AIX su command, and start the BRBACKUP command to back up all tablespaces in offline mode. To start a full offline backup, integrate the following command in your favorite scheduling facility or execute the command from the AIX command line as the AIX root user:

```
su - orac21 -c "brbackup -c -m all -t offline"
```

BRBACKUP automatically:

- Determines the current online redo log number
- Shuts down the database, if the database is online, before starting the backup
- Performs an offline backup of the database, using BACKINT/ADSM
- Performs a backup of the parameter files, using BACKINT/ADSM
- Starts up the database, if the database was online before starting the backup.

SAPDBA performs the preparatory steps for an offline backup before calling BACKINT/ADSM. Figure 133 shows the first part of the resulting backup log file of the full offline backup operation.

```
BRO51I BRBACKUP 3.OF
BRO55I Start of database backup: bcwuepki aff 1 1998-02-25 13.27.36

BR101I Parameters 2

oracle_sid          C21
oracle_home         /oracle/C21
oracle_profile      /oracle/C21/dbs/initC21.ora
sap_profile         /oracle/C21/dbs/initC21.sap
backup_mode         ALL         3
backup_type         offline     4
backup_dev_type     util_file   5
util_par_file       /oracle/C21/dbs/initC21.utl

BR116I ARCHIVE LOG LIST before backup for database instance C21

Database log mode              ARCHIVELOG
Automatic archival             ENABLED
Archive destination            /oracle/C21/saparch/C21arch
Oldest online log sequence     208
Next log sequence to archive   211
Current log sequence           211    SCN: 275640
Database block size            8192

BR118I Tablespaces and data files

PSAPBTABD      ONLINE      ONLINE  /oracle/C21/sapdata4/btabd_1/btabd.data1  340795392  15  3538954  NOLINK  FILE
PSAPBTABI      ONLINE      ONLINE  /oracle/C21/sapdata3/btabi_1/btabi.data1  271589376  10  3670019  NOLINK  FILE
PSAPCLUD       ONLINE      ONLINE  /oracle/C21/sapdata2/clud_1/clud.data1  425730048  6  3604483  NOLINK  FILE
PSAPCLUI       ONLINE      ONLINE  /oracle/C21/sapdata3/clui_1/clui.data1  110108672  11  3670019  NOLINK  FILE
PSAPDDICD      ONLINE      ONLINE  /oracle/C21/sapdata1/ddicd_1/ddicd.data1  85991424  5  3538953  NOLINK  FILE
PSAPDDICI      ONLINE      ONLINE  /oracle/C21/sapdata5/ddici_1/ddici.data1  69214208  22  3604484  NOLINK  FILE
⋮
PSAPTEMP       ONLINE      ONLINE  /oracle/C21/sapdata1/temp_1/temp.data1  104865792  2  3538953  NOLINK  FILE
```

Figure 133 (Part 1 of 2). BRBACKUP Backup Log File for Full Offline Backup Operation

```
PSAPUSER1D     ONLINE      ONLINE  /oracle/C21/sapdata4/user1d_1/user1d.data1  2105344  19  3538954  NOLINK  FILE
PSAPUSER1I     ONLINE      ONLINE  /oracle/C21/sapdata2/user1i_1/user1i.data1  2105344  8  3604483  NOLINK  FILE
SYSTEM         ONLINE      SYSTEM  /oracle/C21/sapdata1/system_1/system.data1  209723392  1  3538953  NOLINK  FILE

BR119I Redo log files

/oracle/C21/origlogA/log_g11m1.dbf  20972032  11  3538951  STALE  NOLINK  FILE
/oracle/C21/mirrlogA/log_g11m2.dbf  20972032  11  3670018  STALE  NOLINK  FILE
/oracle/C21/origlogB/log_g12m1.dbf  20972032  12  3604482  INUSE  NOLINK  FILE
/oracle/C21/mirrlogB/log_g12m2.dbf  20972032  12  3538952  INUSE  NOLINK  FILE
/oracle/C21/origlogA/log_g13m1.dbf  20972032  13  3538951  INUSE  NOLINK  FILE
/oracle/C21/mirrlogA/log_g13m2.dbf  20972032  13  3670018  INUSE  NOLINK  FILE
/oracle/C21/origlogB/log_g14m1.dbf  20972032  14  3604482  INUSE  NOLINK  FILE
/oracle/C21/mirrlogB/log_g14m2.dbf  20972032  14  3538952  INUSE  NOLINK  FILE

BR120I Control files

/oracle/C21/sapdata1/cntrl/cntrlC21.dbf  717312  0  3538953  NOLINK  FILE
/oracle/C21/sapdata2/cntrl/cntrlC21.dbf  717312  0  3604483  NOLINK  FILE
/oracle/C21/dbs/cntrlC21.dbf  717312  0  3538950  NOLINK  FILE

BRO57I Backup of database: C21
BRO58I BRBACKUP action ID: bcwuepki
BRO59I BRBACKUP function ID: aff
BRO60I Tablespaces for backup: ALL
BRO77I Database files for backup:  6
/oracle/C21/origlogA/log_g11m1.dbf
/oracle/C21/origlogB/log_g12m1.dbf
/oracle/C21/origlogA/log_g13m1.dbf
/oracle/C21/origlogB/log_g14m1.dbf
/oracle/C21/sapdata1/cntrl/cntrlC21.dbf
BRO61I 32 files found for backup, total size 6669.897 MB.
BR143I Backup type: offline
BR130I Backup device type: util_file
BR109I Files will be saved by backup utility at file level.
BR126I Unattended mode active - no confirmation required.
```

Figure 133 (Part 2 of 2). BRBACKUP Backup Log File for Full Offline Backup Operation

1 Shows the file name of the backup log file

2 Shows all parameter files used for this backup

3 Indicates that this backup is a full backup including all data files

4 Indicates that this backup is an offline backup

5 Indicates that an external utility such as BACKINT/ADSM will be used to perform the backup

6 Shows the online redo log files and control files that will be saved during an offline backup

Figure 134 shows the part of the resulting backup log file where the BACKINT/ADSM displays the ADSM parameters used for this backup.

```
BR307I Shutting down the database instance C21... 7
BR308I Shutdown of database instance C21 successful.
```

Figure 134 (Part 1 of 2). BRBACKUP Backup Log File for Full Offline Backup Operation: ADSM Parameters

```
BR280I Time stamp 1998-02-25 13.27.51
BR229I Calling backup utility...     8
BR278I Command output of '/usr/sap/C21/SYS/exe/run/backint
-u C21 -f backup -i /oracle/C21/sapbackup/.bcwuepki.lst -t
file -p /oracle/C21/dbs/initC21.utl -c':

   BACKINT/ADSTAR Distributed Storage Manager

  Interface between SAPDBA Utilities and IBM ADSTAR Distributed Storage Manager
  - Version 2, Release 2, Level 5.4 for AIX -
  (c) Copyright IBM Corporation, 1993, 1998, All Rights Reserved.

BKI6001I: The BACKINT/ADSM testing period will expire in 53 days.

  -- Parameters --
Input File         : /oracle/C21/sapbackup/.bcwuepki.lst  9
Output File        : stdout
Profile            : /oracle/C21/dbs/initC21.utl
Configfile         : /oracle/C21/dbs/initC21.bki
Tracefile          : /oracle/C21/dbs/backint_SAP___9802251327.trace
Parallel sessions  : 1       10
Multiplexed files  : 3
RL compression     : enabled
TCPWait            : 1 sec.
Retries to  ADSM   : 3
Retries for files  : 3
Versioning         : enabled
 Current Version   : 2
 Versions to keep  : 4
 Delete Versions   : <= 0
Backup Type        : file
ADSM server        : TERASERV  11  with 1 session configured, using 1 session
 Server-Version    : 3.1.0.1
 Server Host IP    : YELLOW
 Server Host name  : unknown
 Server Type       : AIX-RS/6000
 Sign-in id name   : SJ_C21
 Compression       : client determined
 Archive deletion  : allowed
 Days for backups  : Sun Mon Tue Wed Thu Fri Sat
 Backup mgmt class : R3_DB
 Archiv mgmt class : R3_LOG1 R3_LOG2
```

Figure 134 (Part 2 of 2). BRBACKUP Backup Log File for Full Offline Backup Operation: ADSM Parameters

7 BRBACKUP performs a shutdown of the database if the database was open.

8 BRBACKUP calls BACKINT/ADSM.

9 BRBACKUP compiles a temporary list of files and passes the list to BACKINT/ADSM.

10 These are the parameters of BACKINT/ADSM.

11 Shows the ADSM server with the ADSM parameters BACKINT/ADSM uses to send the data

Figure 135 on page 293 shows the part of the backup log where BRBACKUP calls BACKINT/ADSM and sends the data to ADSM.

```
BKI0005I: Start of backint program at:  12  Wed Feb 25 13:27:51 1998.

BKI0013I: 32 Objects to backup.
BKI0008I: Number of bytes to save: '6.514 GB'.
BKI0027I: Time: 13:27:54 Object: 1 of 32 in process: /oracle/C21/sapdata6/es30fd_1/es30fd.data1
          ... Size: 1492.008 MB. MNGM-Class: R3_DB
BKI0027I: Time: 13:27:54 Object: 2 of 32 in process: /oracle/C21/sapdata2/es30fi_1/es30fi.data1
          ... Size: 799.008 MB. MNGM-Class: R3_DB,
BKI0027I: Time: 13:27:54 Object: 3 of 32 in process: /oracle/C21/sapdata4/el30fd_1/el30fd.data1
          ... Size: 498.008 MB. MNGM-Class: R3_DB,

BR280I Time stamp 1998-02-25 13.52.19
#FILE..... /oracle/C21/sapdata6/es30fd_1/es30fd.data1
#SAVED.... SAP___9802251327
BKI0053I: Time: 13:52:19 Object: 1 of 32 done: /oracle/C21/sapdata6/es30fd_1/es30fd.data1 with 1
BKI2018I: Compression ratio for file '/oracle/C21/sapdata6/es30fd_1/es30fd.data1' was 1.55.
BKI0048I: Object /oracle/C21/sapdata6/es30fd_1/es30fd.data1 with 1492.008 MB saved with
          ... description 'SAP___9802251327_TERASERV@R3_DB
BKI0023I: Time: 13:52:19 Done: 1492.008 MB (22.4%) of 6.514 GB. Estimated end time: 15:17&
BKI0011I: Number of bytes left to be saved: 5.057 GB (77.6%) of 6.514 GB.
   ⋮
BR280I Time stamp 1998-02-25 14.15.38
#FILE..... /oracle/C21/sapdata1/cntrl/cntrlC21.dbf
#SAVED.... SAP___9802251327
BKI0053I: Time: 14:15:38 Object: 32 of 32 done: /oracle/C21/sapdata1/cntrl/cntrlC21.dbf with 700
BKI2018I: Compression ratio for file '/oracle/C21/sapdata1/cntrl/cntrlC21.dbf' was 23.36.
BKI0048I: Object /oracle/C21/sapdata1/cntrl/cntrlC21.dbf with 700.500 KB saved with
          ... description 'SAP___9802251327_TERASERV@R3_DB#011
BKI0023I: Time: 14:15:38 Done: 6.514 GB (100.0%) of 6.514 GB. Estimated end time: 14:15&co
BKI0011I: Number of bytes left to be saved: 0 Bytes (0.0%) of 6.514 GB.

  Nr.  Object                                          Server
   1  /oracle/C21/sapdata6/es30fd_1/es30fd.data1      TERASERV  13
   2  /oracle/C21/sapdata2/es30fi_1/es30fi.data1      TERASERV
   3  /oracle/C21/sapdata4/el30fd_1/el30fd.data1      TERASERV
   4  /oracle/C21/sapdata1/pooli_1/pooli.data1        TERASERV
⋮
  27  /oracle/C21/origlogB/log_g12m1.dbf              TERASERV
  28  /oracle/C21/origlogA/log_g13m1.dbf              TERASERV
  29  /oracle/C21/origlogB/log_g14m1.dbf              TERASERV
  30  /oracle/C21/sapdata4/user1d_1/user1d.data1      TERASERV
  31  /oracle/C21/sapdata2/user1i_1/user1i.data1      TERASERV
  32  /oracle/C21/sapdata1/cntrl/cntrlC21.dbf         TERASERV

  Nr.     Size         Time              Sec    GB/h    Mgmtclass  14
   1   1492.008 MB  00 h 24 min 25 sec   1465   3.580    R3_DB
⋮
  32    700.500 KB  00 h 00 min 05 sec      5   0.481    R3_DB

BKI0020I: End of backint program at: Wed Feb 25 14:15:38 1998.
BKI0022I: Average transmission rate was 8.187 GB/h (2.329 MB/sec).
BKI2019I: Compression ratio for backup was 2.56.
BKI0021I: Elapsed time: 47 min 47 sec.

BR280I Time stamp 1998-02-25 14.15.38
BR232I 32 of 32 files saved by backup utility.
BR230I Backup utility called successfully.

BR304I Starting the database instance C21...    15
BR305I Startup of database instance C21 successful.

BR117I ARCHIVE LOG LIST after backup for database instance C21

Database log mode              ARCHIVELOG
Automatic archival             ENABLED
Archive destination            /oracle/C21/saparch/C21arch
Oldest online log sequence     208
Next log sequence to archive   211
Current log sequence           211    SCN: 275640
Database block size            8192
```

Figure 135. BRBACKUP Backup Log File for Full Offline Backup Operation: ADSM Data Transfer

12 BACKINT/ADSM starts to back up the files to ADSM.

13 This is the summary of the files that BACKINT/ADSM has sent to the ADSM server.

14 BACKINT/ADSM compiles a performance report about each file sent to ADSM.

15 BRBACKUP starts the database because it was in open mode before the backup operation.

Figure 136 shows the final part of the backup log file where BRBACKUP calls BACKINT/ADSM for a second time and BACKINT/ADSM sends the parameter files to ADSM.

```
BR280I Time stamp 1998-02-25 14.16.02
BR229I Calling backup utility...
BR278I Command output of '/usr/sap/C21/SYS/exe/run/backint
-u C21 -f backup -i /oracle/C21/sapbackup/.bcwuepki.lst -t
file -p /oracle/C21/dbs/initC21.utl -c':

               BACKINT/ADSTAR Distributed Storage Manager

  Interface between SAPDBA Utilities and IBM ADSTAR Distributed Storage Manager
            - Version 2, Release 2, Level 5.4 for AIX -
     (c) Copyright IBM Corporation, 1993, 1998, All Rights Reserved.
   ⋮
BKI0005I: Start of backint program at: Wed Feb 25 14:16:02 1998.

BKI0013I: 8 Objects to backup.
BKI0008I: Number of bytes to save: '78.328 KB'.
BKI0027I: Time: 14:16:05 Object: 1 of 8 in process: /oracle/C21/sapbackup/bcwuepki.aff Size&colo

BR280I Time stamp 1998-02-25 14.16.09
#INLOG.... /oracle/C21/sapbackup/bcwuepki.aff
#SAVED.... SAP___9802251416
BKI0053I: Time: 14:16:09 Object: 1 of 8 done: /oracle/C21/sapbackup/bcwuepki.aff with 39.747 KB
BKI0048I: Object /oracle/C21/sapbackup/bcwuepki.aff with 39.747 KB saved with description 'SAP___9802251416_TERASERV@R3_DB#001
BKI0023I: Time: 14:16:09 Done: 39.747 KB (50.7%) of 78.328 KB. Estimated end time: 14:16&c
BKI0011I: Number of bytes left to be saved: 38.581 KB (49.3%) of 78.328 KB.
   ⋮
#INLOG.... /oracle/C21/sapbackup/backC21.log
#SAVED.... SAP___9802251416
BKI0053I: Time: 14:16:51 Object: 8 of 8 done: /oracle/C21/sapbackup/backC21.log with 209 Bytes s
BKI0048I: Object /oracle/C21/sapbackup/backC21.log with 209 Bytes saved with description 'SAP___9802251416_TERASERV@R3_DB#008
BKI0023I: Time: 14:16:51 Done: 78.328 KB (100.0%) of 78.328 KB. Estimated end time: 14:16&
BKI0011I: Number of bytes left to be saved: 0 Bytes (0.0%) of 78.328 KB.

  Nr.  Object                                          Server
   1  /oracle/C21/sapbackup/bcwuepki.aff               TERASERV   16
   2  /oracle/C21/dbs/initC21.dba                      TERASERV
   3  /oracle/C21/dbs/initC21.utl                      TERASERV
   4  /oracle/C21/dbs/initC21.sap                      TERASERV
   5  /oracle/C21/dbs/initC21.ora                      TERASERV
   6  /oracle/C21/sapreorg/structC21.log               TERASERV
   7  /oracle/C21/sapreorg/reorgC21.log                TERASERV
   8  /oracle/C21/sapbackup/backC21.log                TERASERV

  Nr.     Size         Time              Sec    GB/h    Mgmtclass
   1     39.747 KB  00 h 00 min 04 sec     4   0.034    R3_DB
   2     12.106 KB  00 h 00 min 07 sec     7   0.006    R3_DB
   3     10.240 KB  00 h 00 min 06 sec     6   0.006    R3_DB
   4    8955 Bytes  00 h 00 min 06 sec     6   0.005    R3_DB
   5    4498 Bytes  00 h 00 min 04 sec     4   0.004    R3_DB
```

Figure 136 (Part 1 of 2). BRBACKUP Backup Log File for Full Offline Backup Operation: Backup Completion

```
   6    2636 Bytes  00 h 00 min 07 sec      7   0.001    R3_DB
   7     326 Bytes  00 h 00 min 06 sec      6   0.000    R3_DB
   8     209 Bytes  00 h 00 min 06 sec      6   0.000    R3_DB

BKI0020I: End of backint program at: Wed Feb 25 14:16:51 1998.
BKI0022I: Average transmission rate was 0.006 GB/h (0.002 MB/sec).
BKI2019I: Compression ratio for backup was 1.00.
BKI0021I: Elapsed time: 49 sec.

BR280I Time stamp 1998-02-25 14.16.51
BR232I 8 of 8 files saved by backup utility.
BR230I Backup utility called successfully.

BR056I End of database backup: bcwuepki aff  1998-02-25 14.16.51
BR052I BRBACKUP terminated successfully.
```

Figure 136 (Part 2 of 2). BRBACKUP Backup Log File for Full Offline Backup Operation: Backup Completion

16 BACKINT/ADSM saves the parameter files and backup log files within the same backup ID.

10.5.2 Full Online Backup

A full online backup is most often performed if the database must be online all the time.

Switch to the Oracle orac21 user, using the AIX su command, and start the BRBACKUP command to back up the database in online mode. From the AIX command line as the AIX orac21 user:

```
su - orac21 -c "brbackup -c -m all -t online"
```

Figure 137 shows the first part of the resulting backup log file of the full online backup operation.

```
BR051I BRBACKUP 3.0F
BR055I Start of database backup: bcwuezlq anf  1998-02-25 15.20.50

BR319I Control file copied to /oracle/C21/sapbackup/cntrl.dbf 1
BR101I Parameters
oracle_sid          C21
oracle_home         /oracle/C21
oracle_profile      /oracle/C21/dbs/initC21.ora
sap_profile         /oracle/C21/dbs/initC21.sap
backup_mode         ALL
backup_type         online  2
backup_dev_type     util_file  3
util_par_file       /oracle/C21/dbs/initC21.utl

BR116I ARCHIVE LOG LIST before backup for database instance C21

Database log mode              ARCHIVELOG
Automatic archival             ENABLED
Archive destination            /oracle/C21/saparch/C21arch
Oldest online log sequence     208
Next log sequence to archive   211
```

Figure 137 (Part 1 of 2). BRBACKUP Backup Log File from Full Online Backup Operation

```
Current log sequence           211    SCN: 275640
Database block size            8192
```

Figure 137 (Part 2 of 2). BRBACKUP Backup Log File from Full Online Backup Operation

1 BRBACKUP executes an Oracle internal command to create a read-consistent version of the /oracle/C21/sapbackup/cntrl.dbf control file

2 Specifies that BRBACKUP performs an online backup of the database

3 Indicates that BRBACKUP calls BACKINT/ADSM

Figure 138 shows the second part of the backup log file of the full online backup operation.

```
BR118I Tablespaces and data files

PSAPBTABD      ONLINE      ONLINE  /oracle/C21/sapdata4/btabd_1/btabd.data1  340795392  15  3538954  NOLINK  FILE
PSAPBTABI      ONLINE      ONLINE  /oracle/C21/sapdata3/btabi_1/btabi.data1  271589376  10  3670019  NOLINK  FILE
PSAPCLUD       ONLINE      ONLINE  /oracle/C21/sapdata2/clud_1/clud.data1  425730048  6  3604483  NOLINK  FILE
⋮
SYSTEM         ONLINE      SYSTEM  /oracle/C21/sapdata1/system_1/system.data1  209723392  1  3538953  NOLINK  FILE

BR119I Redo log files

/oracle/C21/origlogA/log_g11m1.dbf  20972032  11  3538951  STALE  NOLINK  FILE
/oracle/C21/mirrlogA/log_g11m2.dbf  20972032  11  3670018  STALE  NOLINK  FILE
/oracle/C21/origlogB/log_g12m1.dbf  20972032  12  3604482  INUSE  NOLINK  FILE
/oracle/C21/mirrlogB/log_g12m2.dbf  20972032  12  3538952  INUSE  NOLINK  FILE
/oracle/C21/origlogA/log_g13m1.dbf  20972032  13  3538951  INUSE  NOLINK  FILE
/oracle/C21/mirrlogA/log_g13m2.dbf  20972032  13  3670018  INUSE  NOLINK  FILE
/oracle/C21/origlogB/log_g14m1.dbf  20972032  14  3604482  INUSE  NOLINK  FILE
/oracle/C21/mirrlogB/log_g14m2.dbf  20972032  14  3538952  INUSE  NOLINK  FILE

BR120I Control files

/oracle/C21/sapdata1/cntrl/cntrlC21.dbf  717312  0  3538953  NOLINK  FILE
/oracle/C21/sapdata2/cntrl/cntrlC21.dbf  717312  0  3604483  NOLINK  FILE
/oracle/C21/dbs/cntrlC21.dbf  717312  0  3538950  NOLINK  FILE

BRO57I Backup of database: C21
BRO58I BRBACKUP action ID: bcwuezlq
BRO59I BRBACKUP function ID: anf
BRO60I Tablespaces for backup: ALL
BRO77I Database file for backup: /oracle/C21/sapbackup/cntrl.dbf 4
BRO61I 28 files found for backup, total size 6589.895 MB.
BR143I Backup type: online
BR130I Backup device type: util_file
BR109I Files will be saved by backup utility at file level.
BR126I Unattended mode active - no confirmation required.

BR315I Alter tablespace PSAPBTABD begin backup successful. 5

BR315I Alter tablespace PSAPBTABI begin backup successful.
⋮
BR315I Alter tablespace SYSTEM begin backup successful.
```

Figure 138. BRBACKUP Backup Log File from Full Online Backup Operation: Backup Initiation

4 Because the primary control file is in use during an online backup operation, this backup control file will be sent to ADSM.

5 BRBACKUP executes an Oracle internal command to alter the tablespaces for online backup.

Figure 139 on page 297 shows the final part of the backup log file of the full online backup operation.

```
BR229I Calling backup utility...
BR278I Command output of '/usr/sap/C21/SYS/exe/run/backint
-u C21 -f backup -i /oracle/C21/sapbackup/.bcwuezlq.lst
-t file -p /oracle/C21/dbs/initC21.utl -c':
⋮
BKI0020I: End of backint program at: Wed Feb 25 16:03:59 1998.
BKI0022I: Average transmission rate was 9.039 GB/h (2.571 MB/sec).
BKI2019I: Compression ratio for backup was 2.56.
BKI0021I: Elapsed time: 42 min 47 sec.

BR280I Time stamp 1998-02-25 16.03.59
BR232I 28 of 28 files saved by backup utility.
BR230I Backup utility called successfully.

BR317I Alter tablespace PSAPBTABD end backup successful. 6

BR317I Alter tablespace PSAPBTABI end backup successful.

BR317I Alter tablespace PSAPCLUD end backup successful.
⋮
BR317I Alter tablespace PSAPUSER1D end backup successful.

BR317I Alter tablespace PSAPUSER1I end backup successful.

BR317I Alter tablespace SYSTEM end backup successful.

BR340I Switching to next online redo log file for database instance C21. 7
BR321I Switch to next online redo log file for database instance C21 successful.


BR117I ARCHIVE LOG LIST after backup for database instance C21

Database log mode              ARCHIVELOG
Automatic archival             ENABLED
Archive destination            /oracle/C21/saparch/C21arch
Oldest online log sequence     209
Next log sequence to archive   211
Current log sequence           212    SCN: 276105
Database block size            8192

BR280I Time stamp 1998-02-25 16.04.14
BR229I Calling backup utility...
⋮
BR230I Backup utility called successfully.

BR056I End of database backup: bcwuezlq anf  1998-02-25 16.05.03
BR052I BRBACKUP terminated successfully.
```

Figure 139. BRBACKUP Backup Log File from Full Online Backup Operation: Backup Completion

6 BRBACKUP executes an Oracle internal command to alter the tablespaces for online backup (END BACKUP).

7 BRBACKUP executes an Oracle internal command to switch the current online redo log file. The next online redo log file will be used.

The full online backup we present is one way of performing a full online backup. There is another way of performing online backups with BRBACKUP. You can call the BRBACKUP command with the -t util_file_online parameter. This parameter indicates that BRBACKUP executes the Oracle internal *ALTER TABLESPACE BEGIN BACKUP* and *ALTER TABLESPACE END BACKUP* commands for every tablespace exactly when BACKINT/ADSM is about to read the tablespace for backup. If you call BRBACKUP with the -t util_file_online parameter, BACKINT/ADSM knows about that and needs to synchronize its

operations with BRBACKUP. BRBACKUP then calls BACKINT/ADSM with the -t file_online option, which tells BACKINT/ADSM to synchronize with BRBACKUP for every tablespace altered in backup mode.

Figure 140 shows a different backup log file of a separate full online backup operation using the BRBACKUP -t util_file_online option.

```
BR101I Parameters

oracle_sid          C21
oracle_home         /oracle/C21
oracle_profile      /oracle/C21/dbs/initC21.ora
sap_profile         /oracle/C21/dbs/initC21.sap
backup_mode         ALL
backup_type         online
backup_dev_type     util_file_online
util_par_file       /oracle/C21/dbs/initC21.utl
⋮
BR130I Backup device type: util_file_online 8
BR109I Files will be saved by backup utility at file level.
BR142I Files will be switched to backup status during the backup.
BR126I Unattended mode active - no confirmation required.

BR280I Time stamp 1998-02-25 18.30.02
BR229I Calling backup utility...
BR278I Command output of '/usr/sap/C21/SYS/exe/run/backint
-u C21 -f backup -i /oracle/C21/sapbackup/.bcwufqge.lst -t file_online
-p /oracle/C21/dbs/initC21.utl -c':

BKI2012I: Multiplexing is not supported in 'util_file_online' mode. 9
⋮
BKI0013I: 28 Objects to backup.
BKI0008I: Number of bytes to save: '6.435 GB'.
BR280I Time stamp 1998-02-25 18.30.07
#BEGIN /oracle/C21/sapdata6/es30fd_1/es30fd.data1
BR315I Alter tablespace PSAPES30FD begin backup successful. 10
BR280I Time stamp 1998-02-25 18.30.08
BKI0027I: Time: 18:30:09 Object: 1 of 28 in process: /oracle/C21/sapdata6/es30fd_1/es30fd.data1
        ...  Size: 1492.008 MB. MNGM-Class: R3_DB
BR280I Time stamp 1998-02-25 18.45.28
#END /oracle/C21/sapdata6/es30fd_1/es30fd.data1
BR317I Alter tablespace PSAPES30FD end backup successful. 11
BR280I Time stamp 1998-02-25 18.45.30
⋮
BR056I End of database backup: bcwufqge anf  1998-02-25 19.19.11
BR052I BRBACKUP terminated successfully.
```

Figure 140. SAP R/3 BRBACKUP Backup Log File from Full Online Backup Operation Using util_file_online

8 Indicates that BRBACKUP will use the util_file_online parameter

9 Multiplexing is not supported. BRBACKUP reads one tablespace at a time, so multiplexing of tablespaces does not take effect.

10 BRBACKUP executes the Oracle *ALTER TABLESPACE BEGIN BACKUP* command for each tablespace before the file is backed up by BACKINT/ADSM.

11 BRBACKUP executes the Oracle *ALTER TABLESPACE END BACKUP* command after BACKINT/ADSM has backed up the data files of the tablespace.

10.5.3 Backup of Archived Redo Logs

You must save the archived redo logs for every type of backup: offline, online, full, or partial. This is the only way of recovering the database to a specific point in time.

SAP recommends that you use the BRARCHIVE command to copy the logs twice to the ADSM server and, whenever possible, to different tapes. BRARCHIVE deletes the logs from file system storage after completing the archive of the second copy.

We recommend that you perform the archive process of redo logs as often as possible to prevent the loss of any archived redo logs.

To back up the archived redo logs:

1. Start SAPDBA.
2. On the *SAPDBA V3.0F - SAP Database Administration* menu, select option **i** to back up the archived redo logs. The *Backup archive logs* menu is displayed, as shown in Figure 141.

```
                          Backup archive logs

                                              Current value
  a - Archive Function                        Save archive logs
  b - Parameter file                          initC21.sap
  c - Confirm archive parameters              no
  d - Language                                English
  e - Enter archive device type               util_file
  f - Backup utility parameter file           /oracle/C21/dbs/initC21.utl
  g - Number of redo logs                     10000
  h - Enter password interactively            no
  i - Query only                              no
  m - Continue until end of tape              no

  s - Start BRARCHIVE
  q - Return

  Please select  ==> s
```

Figure 141. SAPDBA Backup Archive Logs Menu

3. Start the backup archive logs operation by entering option **s** and pressing Enter.

   The BRARCHIVE command to perform a backup of the archived redo logs from the AIX command line as user *orac21*, accepting the default values as defined in */oracle/C21/dbs/initC21.sap*, is:

   ```
   brarchive -c -sd
   ```

   The BRARCHIVE command logs the results in the */oracle/C21/saparch/acwudwwa.svd.* file. Figure 142 on page 300 shows first part of the resulting archive log file.

```
BR002I BRARCHIVE 3.0F
BR006I Start of offline redo log processing: acwudwwa svd  1998-02-25 09.58.36

BR101I Parameters

oracle_sid          C21
oracle_home         /oracle/C21
oracle_profile      /oracle/C21/dbs/initC21.ora
sap_profile         /oracle/C21/dbs/initC21.sap
backup_dev_type     util_file
util_par_file       /oracle/C21/dbs/initC21.utl

BR008I Offline redo log processing for database instance: C21
BR009I BRARCHIVE action ID: acwudwwa
BR010I BRARCHIVE function ID: svd
BR011I 3 offline redo log files found for processing, total size 11.290 MB.  1
BR130I Backup device type: util_file
BR109I Files will be saved by backup utility at file level.
BR126I Unattended mode active - no confirmation required.

BR280I Time stamp 1998-02-25 09.58.37
BR229I Calling backup utility...
BR278I Command output of '/usr/sap/C21/SYS/exe/run/backint
-u C21 -f backup -i /oracle/C21/sapbackup/.acwudwwa.lst
10
⋮
BKI0005I: Start of backint program at: Wed Feb 25 09:58:37 1998.

BKI0013I: 6 Objects to backup.  2
BKI0008I: Number of bytes to save: '22.580 MB'.
BKI0026I: Time: 09:58:40 Object: 1 of 6 in process: /oracle/C21/saparch/C21arch1_206.dbf
          ... Size: 11.239 MB. MNGM-Class: R3_LOG1, ADSM
BKI0053I: Time: 09:58:54 Object: 1 of 6 done: /oracle/C21/saparch/C21arch1_206.dbf with 11.239 M
          ... saved with description 'SAP___9802250958_TERASERV@R3_LOG1#001 1
BKI0048I: Object /oracle/C21/saparch/C21arch1_206.dbf with 11.239 MB saved with
          ... description 'SAP___9802250958_TERASERV@R3_LOG1#001 1
BKI0023I: Time: 09:58:54 Done: 11.239 MB (49.8%) of 22.580 MB. Estimated end time: 09:59&c
BKI0011I: Number of bytes left to be saved: 11.341 MB (50.2%) of 22.580 MB.
⋮
BR280I Time stamp 1998-02-25 09.59.16
#ARCHIVE.. /oracle/C21/saparch/C21arch1_208.dbf
#SAVED.... SAP___9802250958
BKI0053I: Time: 09:59:16 Object: 6 of 6 done: /oracle/C21/saparch/C21arch1_208.dbf with 40.500 K
          ... saved with description 'SAP___9802250958_TERASERV@R3_LOG2#006 2
BKI0048I: Object /oracle/C21/saparch/C21arch1_208.dbf with 40.500 KB saved with
          ... description 'SAP___9802250958_TERASERV@R3_LOG2#006 2
BKI0023I: Time: 09:59:16 Done: 22.580 MB (100.0%) of 22.580 MB. Estimated end time: 09:59&
BKI0011I: Number of bytes left to be saved: 0 Bytes (0.0%) of 22.580 MB.
```

Figure 142 (Part 1 of 2). BRARCHIVE Archive of Log File: Archive Initiation

```
  Nr.  Object                                          Server
   1  /oracle/C21/saparch/C21arch1_206.dbf             TERASERV
   2  /oracle/C21/saparch/C21arch1_206.dbf             TERASERV
   3  /oracle/C21/saparch/C21arch1_207.dbf             TERASERV
   4  /oracle/C21/saparch/C21arch1_207.dbf             TERASERV
   5  /oracle/C21/saparch/C21arch1_208.dbf             TERASERV
   6  /oracle/C21/saparch/C21arch1_208.dbf             TERASERV
⋮

BKI0020I: End of backint program at: Wed Feb 25 09:59:16 1998.
BKI0022I: Average transmission rate was 2.205 GB/h (0.627 MB/sec).
BKI2019I: Compression ratio for backup was 1.00.
BKI0021I: Elapsed time: 39 sec.
BKI0100I: 2 Copies of each file saved. 3

BR280I Time stamp 1998-02-25 09.59.16
BR232I 3 of 3 files saved by backup utility.
BR230I Backup utility called successfully.

BR015I Offline redo log file /oracle/C21/saparch/C21arch1_206.dbf deleted. 4
BR015I Offline redo log file /oracle/C21/saparch/C21arch1_207.dbf deleted.
BR015I Offline redo log file /oracle/C21/saparch/C21arch1_208.dbf deleted.

BR016I 3 offline redo log files processed.
```

Figure 142 (Part 2 of 2). BRARCHIVE Archive of Log File: Archive Initiation

1 BRARCHIVE has found three offline redo logs on disk that need a backup.

2 BRARCHIVE will send six objects to ADSM because each object is backed up twice for disaster recovery purposes.

3 BACKINT/ADSM has saved two copies of each file (on different media if possible).

4 BRARCHIVE deletes the successfully saved offline redo logs from disk. Figure 143 shows the last part of the resulting archive log file.

```
BR280I Time stamp 1998-02-25 09.59.17
BR229I Calling backup utility...
BR278I Command output of '/usr/sap/C21/SYS/exe/run/backint -u C21 -f
       ... backup -i /oracle/C21/sapbackup/.acwudwwa.lst -t file -p /oracl
⋮
  Nr.  Object  5                                    Server
   1  /oracle/C21/dbs/initC21.ora                        TERASERV
   2  /oracle/C21/dbs/initC21.ora                        TERASERV
   3  /oracle/C21/dbs/initC21.dba                        TERASERV
   4  /oracle/C21/dbs/initC21.dba                        TERASERV
   5  /oracle/C21/dbs/initC21.sap                        TERASERV
   6  /oracle/C21/dbs/initC21.sap                        TERASERV
   7  /oracle/C21/dbs/initC21.utl                        TERASERV
   8  /oracle/C21/dbs/initC21.utl                        TERASERV
   9  /oracle/C21/sapreorg/reorgC21.log                  TERASERV
  10  /oracle/C21/sapreorg/reorgC21.log                  TERASERV
  11  /oracle/C21/sapreorg/structC21.log                 TERASERV
  12  /oracle/C21/sapreorg/structC21.log                 TERASERV
  13  /oracle/C21/saparch/acwudwwa.svd                   TERASERV
  14  /oracle/C21/saparch/acwudwwa.svd                   TERASERV
```

Figure 143 (Part 1 of 2). BRARCHIVE Archive of Log File: Archive Completion

```
  15  /oracle/C21/saparch/archC21.log                    TERASERV
  16  /oracle/C21/saparch/archC21.log                    TERASERV
⋮

BKI0020I: End of backint program at: Wed Feb 25 09:59:53 1998.
BKI0022I: Average transmission rate was 0.010 GB/h (0.003 MB/sec).
BKI2019I: Compression ratio for backup was 1.00.
BKI0021I: Elapsed time: 36 sec.
BKI0100I: 2 Copies of each file saved.

BR280I Time stamp 1998-02-25 09.59.53
BR232I 8 of 8 files saved by backup utility.
BR230I Backup utility called successfully.

BR007I End of offline redo log processing: acwudwwa svd  1998-02-25 09.59.53
BR003I BRARCHIVE terminated successfully.
```

Figure 143 (Part 2 of 2). BRARCHIVE Archive of Log File: Archive Completion

5 BRARCHIVE backs up all necessary parameter files and backup logs and saves two copies of each file.

10.5.4 Managing Backups with Backfm

The BACKINT/ADSM tool *backfm* helps you display, restore, or delete files within specific backup IDs. You use *backfm* when you want to know which files are stored within a backup ID. Since BACKINT/ADSM Version 2 uses the ADSM API and does not use any description strings of the ADSM dsmc archive command, *backfm* is a helpful tool to see which files a backup ID contains. It is easier to use than issuing ADSM queries to the ADSM server, because it just displays the relevant files for SAP R/3. Backfm queries the ADSM server based on the backup ID.

You invoke *backfm* from the AIX command line as the Oracle administrative user, in our examples, *orac21*: Backfm needs the BACKINT/ADSM utility parameter file to be specified with the -p option:

```
backfm -p /oracle/C21/dbs/initC21.utl
```

Backfm checks all valid backup log files stored in */oracle/C21/sapbackup/backC21.log* and analyzes the contents of each backup log file. Figure 144 on page 303 shows the main menu of the *backfm* utility where the backup IDs that contain all backups taken, even unsuccessful backups, are displayed.

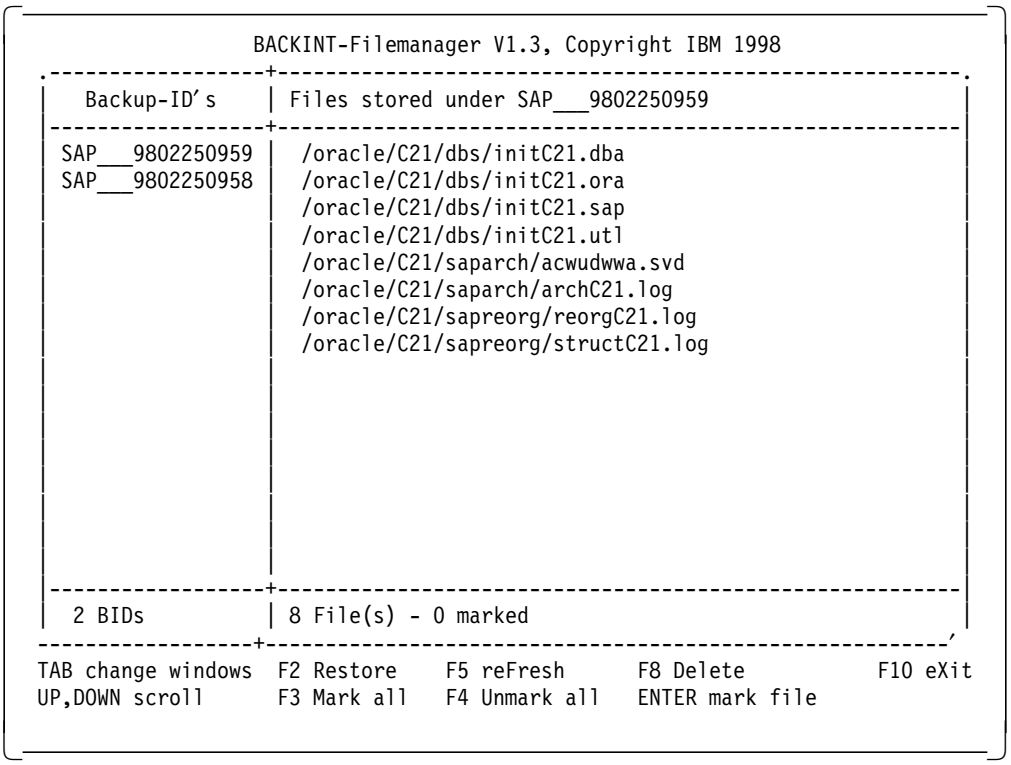


Figure 144. Displaying Backups IDs and Files with backfm

On the left side, backfm displays all backup IDs, and on the right side, all files that are included within a backup ID. You can use the function keys to select and restore specific files or even an entire backup id.

In our example we select backup ID **SAP___9802250959** and restore one file called **/oracle/C21/saparch/archC21.log**.

Follow these steps to restore the file:

1. With the cursor key, go down to the file you want to restore.
2. Press Enter to mark the file (an asterisk (*****) appears in front of the file).
3. Press function key **F2** for restore.
4. Optionally enter a new destination directory.
5. Press enter to perform the operation.

Figure 145 on page 304 shows the output of the restore operation with backfm:

```
BKI0014I: 1 Objects to restore.
BKI0009I: Number of bytes to restore: '900 Bytes'.
BKI0030I: Time: 14:20:22 Object: 1 of 1 in process: /oracle/C21/saparch/archC21.
log Size: 900 Bytes. ADSM server: TERASERV.
#RESTORED SAP___9802250959 /oracle/C21/saparch/archC21.log 1
BKI0054I: Time: 14:20:23 Object: 1 of 1 done: /oracle/C21/saparch/archC21.log
 ...with 900 Bytes restored with description 'SAP___9802250959'
BKI0049I: Object /oracle/C21/saparch/archC21.log with 900 Bytes restored with
 ...description 'SAP___9802250959_TERASERV@R3_LOG1#0081.COPY 0001'
BKI0023I: Time: 14:20:23 Done: 900 Bytes (100.0) of 900 Bytes. Estimated end
 ...time: 14:20:23.
BKI0012I: Number of bytes left to be restored : 0 Bytes (0.0) of 900 Bytes.

  Nr.  Object                                              Server
   1  /oracle/C21/saparch/archC21.log                     TERASERV

  Nr.     Size         Time               Sec    GB/h    Mgmtclass
   1     900 Bytes  00 h 00 min 01 sec      1   0.003    R3_LOG1

BKI0020I: End of backint program at: Wed Feb 25 14:20:23 1998.
BKI0022I: Average transmission rate was 0.003 GB/h (0.001 MB/sec).
BKI0021I: Elapsed time: 03 sec.
```

Figure 145. Output from backfm Restoring a File

1 The backup ID and the file name will be restored.

10.5.5 Restore and Recover a Data File

This example shows how to restore a single data file of the Oracle database. There are many reasons why a data file can be lost:

- The data file has been deleted inadvertently by a user.
- The data file became lost because of a media error.
- The data file became corrupt because of other improper database actions performed by a DBA.

In most cases a data file becomes lost when the database is online, and in this case the DBA has to restore it and perform recovery. To restore and recover a data file to its most current state, follow these steps:

1. Shut down the database:

   ```
   sapdba -shutdown
   ```

2. Start SAPDBA:

   ```
   sapdba
   ```

3. At the SAPDBA menu (Figure 129 on page 282):

   - Select option **m** to identify yourself as the expert.
   - Select option **k**, Restore/Recovery.

   The SAPDBA Restore / Recovery menu is displayed (Figure 146 on page 305).

```
                          Restore / recovery

                a  -  Full restore
                      (incl. redo logs and control files)
                b  -  Full restore and recovery
                      (excl. redo logs)

                c  -  Restore one tablespace
                d  -  Restore individual file(s)

                q  -  Return

                Please select  ==> d
```

Figure 146. SAPDBA Restore / Recovery Menu

4. Select option **d**, Restore individual file(s), and press Enter.

   The SAPDBA *Restore individual file(s)* menu is displayed (Figure 147).

```
                       Restore individual file(s)

                A  -  File Type        :<no selection>
                B  -  BRBACKUP Run     :<no selection>
                C  -  Enter/Show Files :<no selection>

                q  -  Return

                Please select  ==> A
```

Figure 147. SAPDBA Restore Individual File(s) Menu

5. Select option **A** to specify the file type and press Enter.

   The SAPDBA Select file type for restore menu is displayed (Figure 148 on page 306).

```
                        Select file type for restore

       NOTE: Profiles and log files can only be restored from a tape!

          a -  Non-database files
       -> b -  Data files
          c -  Control file
          d -  Online redo logs
          e -  Offline redo logs
          f -  ORACLE and SAPDBA profiles
          g -  BRBACKUP/BRARCHIVE profile (init.sap)
          h -  SAPDBA main and struct log file
          i -  Summary BRBACKUP/BRARCHIVE log file
          j -  Detail BRBACKUP/BRARCHIVE log file

          q -  Return

          Please select  ==>
```

Figure 148. SAPDBA Select File Type for Restore Menu

6. Select option **b** to display the BRBACKUP log file that will be used to restore the data file.

   SAPDBA queries the master backup log file, *backC21.log*, about which backup log files are available and displays them:

```
   1 - bcwuepki aff 1998-02-25 13.27.36    2 - bcwuezlq anf 1998-02-25 15.20.50
   3 - bcwufqge anf 1998-02-25 18.30.00
```

   By default the last backup log file, which is the most current file, will be selected.

7. Select the appropriate backup log file from which you want to restore the data file:

```
Please, enter the number of the BRBACKUP run from that you want to restore
(q = quit) [3] ==> 3
Selected BRBACKUP run:   action id: bcwufqge
                        function id: anf
```

   The SAPDBA *Restore individual file(s)* menu (Figure 147 on page 305) is displayed again.

8. Select option **C**, Enter/Show Files, to display all files belonging to the selected backup log file.

9. Select the file you want to restore by entering the number of the data file. The SAPDBA *Restore individual file(s)* menu is displayed.

10. Select option **S**, Start BRRESTORE, to start the restore operation. BRRESTORE calls BACKINT/ADSM and displays the parameters used for restore as shown in Figure 149 on page 307.

```
                        Specify restore parameters

                                          Current value
     a  -  BRBACKUP profile               initC21.sap
     e  -  Backup utility parameter file  /oracle/C21/dbs/initC21.utl
     g  -  Language                       English

     q  -  Return to Restore individual file(s) and continue
     r  -  Cancel

     Please select  ==> q

List of restore parameters
```

Figure 149. SAPDBA Specify Restore Parameters Menu

11. BRRESTORE calls BACKINT/ADSM with the specified parameters (Figure 150).

```
BRBACKUP profile             initC21.sap
Backup utility parameter file /oracle/C21/dbs/initC21.utl
Language                     English

SAPDBA: SAPDBA does not check whether the database is up or down. It is your
        responsibility to take the database into the correct state.

SAPDBA: After restoring individual files you will probably need to recover your
        database.

SAPDBA: SAPDBA does not check whether it makes sense to restore the file(s) or
        not.

Do you want to continue? (y/n) [y] ==>
BR401I BRRESTORE 3.0F
BR169I Value 'util_file_online' of parameter backup_dev_type ignored for
       ... 'brrestore' - 'util_file' assumed.
BR405I Start of database restore: rcwujsel rsb  1998-02-26 14.23.31
BR457W Probable the database must be recovered due to partial restore!
BR256I Please enter 'cont' to continue, 'stop' to cancel the program:
```

Figure 150. BRRESTORE Calls BACKINT/ADSM with the Specified Parameters

12. Continue by entering **cont** on the interactive SAPDBA screen. BACKINT/ADSM restores the specified data file from ADSM. Figure 151 shows the output of the BACKINT/ADSM restore operation.

```
BR407I Restore of database: C21
BR408I BRRESTORE action ID: rcwujsel
BR409I BRRESTORE function ID: rsb
BR411I Database file for restore: /oracle/C21/sapdata4/user1d_1/user1d.data1 1
BR419I Files will be restored from backup: bcwufqge.anf 1998-02-25 18.30.00  2
BR416I 1 file found to restore, size 2.008 MB.
BR421I Backup device type for restore: util_file
BR256I Please enter 'cont' to continue, 'stop' to cancel the program:
```

Figure 151 (Part 1 of 2). BACKINT/ADSM Data File Restore Output

```
cont
BR257I Your reply: 'cont'
BR259I Program execution will be continued...

BR280I Time stamp 1998-02-26 14.24.04
BR229I Calling backup utility...
BR278I Command output of '/usr/sap/C21/SYS/exe/run/backint  3
-u C21 -f restore -i /oracle/C21/sapbackup/.rcwujsel.lst -t file -p
/oracle/C21/dbs/initC21.utl':

                   BACKINT/ADSTAR Distributed Storage Manager
⋮
BKI0005I: Start of backint program at: Thu Feb 26 14:24:04 1998.

BKI0014I: 1 Objects to restore.
BKI0009I: Number of bytes to restore: '2056.000 KB'.
BKI0030I: Time: 14:24:06 Object: 1 of 1 in process: /oracle/C21/sapdata4/user1d_1/user1d.data1
          ... Size: 2056.000 KB. ADSM server: TERASERV.

BR280I Time stamp 1998-02-26 14.27.13
#FILE..... /oracle/C21/sapdata4/user1d_1/user1d.data1
#RESTORED. SAP___9802251830
BKI0054I: Time: 14:27:13 Object: 1 of 1 done: /oracle/C21/sapdata4/user1d_1/user1d.data1
      ... with 2056.000 KB restored with description 'SAP___9802251830'
BKI0049I: Object /oracle/C21/sapdata4/user1d_1/user1d.data1 with 2056.000 KB restored with
      ... description 'SAP___9802251830_TERASERV@R3_DB#026 0004'
BKI0023I: Time: 14:27:13 Done: 2056.000 KB (100.0%) of 2056.000 KB. Estimated end time:
14:27:13.
BKI0012I: Number of bytes left to be restored : 0 Bytes (0.0%) of 2056.000 KB.

  Nr.  Object                                          Server
   1  /oracle/C21/sapdata4/user1d_1/user1d.data1       TERASERV    4

  Nr.     Size         Time              Sec    GB/h    Mgmtclass
   1   2056.000 KB  00 h 03 min 07 sec    187   0.038    R3_DB

BKI0020I: End of backint program at: Thu Feb 26 14:27:13 1998.
BKI0022I: Average transmission rate was 0.038 GB/h (0.011 MB/sec).
BKI0021I: Elapsed time: 03 min 09 sec.

BR280I Time stamp 1998-02-26 14.27.13
BR374I 1 of 1 file restored by backup utility.
BR230I Backup utility called successfully.
⋮
SAPDBA: Restored file "/oracle/C21/sapdata4/user1d_1/user1d.data1"
        successfully.

SAPDBA: Restore files: 1 file(s) successful,
                       0 file(s) with errors
```

Figure 151 (Part 2 of 2). BACKINT/ADSM Data File Restore Output

1 BRBACKUP identifies the file that needs to be restored.

2 This is the backup log file from which the file will be restored.

3 BRBACKUP calls BACKINT/ADSM to restore the data file.

4 BACKINT/ADSM shows the file that has been restored from the ADSM server. Here you can see whether different ADSM servers are used for the restore.

The data file has been restored properly. You now need to perform the SAPDBA check and repair database functions to check whether the data file needs recovery.

13. Go back to the main SAPDBA menu by entering **Q** for quit.

14. Select option **J**, Check (and Repair) database, from the SAPDBA main menu to display the SAPDBA Check (and repair) database menu (Figure 152 on page 309).

```
                    Check (and repair) database

                                                Status

          a  -  Check   database                not finished
          b  -  Find    backup  files           not finished
          c  -  Restore backup  files           not finished
          d  -  Find    archive files           not finished
          e  -  Restore archive files           not finished
          f  -  Recover database                not finished

          g  -  Automatic recovery

          q  -  Return


          Please select  ==> g
```

Figure 152. SAPDBA Check (and Repair) Database Menu

15. Select option **g**, Automatic recovery. SAPDBA checks the entire database by inspecting all database objects. Figure 153 shows how SAPDBA checks the Oracle database objects.

```
*************** SAPDBA - Checking database status via V$-tables ***************

****************** SAPDBA - Checking control files on disk ******************  5

1.Controlfile '/oracle/C21/sapdata1/cntrl/cntrlC21.dbf' is on disk
2.Controlfile '/oracle/C21/sapdata2/cntrl/cntrlC21.dbf' is on disk
3.Controlfile '/oracle/C21/dbs/cntrlC21.dbf' is on disk

SAPDBA: All control files are on disk.
        Database mounted

Press <return> to continue ...

*********** SAPDBA - Checking data file type and structure on disk ***********

SAPDBA:   27 files exist according to V$DATAFILE.

SAPDBA: All   27 files are regular files.

Press <return> to continue ...

******************** SAPDBA - Checking data files on disk ********************

*************** SAPDBA - Summary: Checking data files on disks ***************

SAPDBA: All data files are on disk.
```

Figure 153 (Part 1 of 2). SAPDBA Messages before Database Recovery

```
Press <return> to continue ...
⋮
Do you want a safe check? (y/n) [n] ==> y    6

SAPDBA: To check the database it has to be closed
        and mounted again
        (safe check)

SAPDBA: Database mounted and will be shut down immediately.
        Do you want to continue?

Please select (y/n) [y] ==>
⋮
SAPDBA: Instance C21 successfully shut down.
⋮
SAPDBA: Instance C21 mounted.
SAPDBA: Database mounted
⋮
SAPDBA: All control files are O.K.
        Database mounted
```

Figure 153 (Part 2 of 2). SAPDBA Messages before Database Recovery

5 SAPDBA performs a check of the database if all objects have the correct state and are on their correct destination directories.

6 SAPDBA performs a safe check (a more detailed check) of the database objects. Therefore SAPDBA has to shut down and restart the database.

SAPDBA now performs automatic recovery of the Oracle database. Oracle applies all necessary archive redo logs to bring to a consistent state the tablespace to which the restored data file belongs, as shown in Figure 154.

```
*************** SAPDBA - Checking backup status of tablespaces ***************

SAPDBA: The following data file(s) are still in backup mode:

        Data file '/oracle/C21/sapdata4/user1d_1/user1d.data1'
SAPDBA: 1 file is active (to be backed up).  7
SAPDBA: You should check the status of the related tablespaces
        by using this function after having the database opened!
        If you have restored files from an online backup
        the status 'active' is what you expect

Press <return> to continue ...

********* SAPDBA - Checking data file's ONLINE status via V$DATAFILE *********

SAPDBA: All data files are online.

Press <return> to continue ...

****** SAPDBA - Checking data file's RECOVERY status via V$RECOVER_FILE ******

SAPDBA: Data file '/oracle/C21/sapdata4/user1d_1/user1d.data1'
        needs recovery from 212 to 213 for thread 1     8
        but no restore

Do you rather want to restore a newer data file? (y/n) [n] ==> n    9

Do you want to skip this dialog for all further existing datafiles (Always recovery)?
```

Figure 154 (Part 1 of 2). SAPDBA Messages during Database Recovery

```
(y/n) [n] ==> y
⋮
************************* SAPDBA - Recover database *************************

SAPDBA: Recovery requires online redo logs only   10
        beginning from 212

Press <return> to continue ...

SAPDBA: Database successfully recovered using online redo logs.

SAPDBA: Trying to startup database from mount to open ...

SAPDBA: Database mounted and opened.

Press <return> to continue ...

SAPDBA: Database open

* * * * * * * * * * * * * * * * * * * * * * * * * * * *
* SAPDBA: Recover database terminated successfully. *
* * * * * * * * * * * * * * * * * * * * * * * * * * * *
Press <return> to continue ...

Automatic recovery terminated successfully.
```

Figure 154 (Part 2 of 2). SAPDBA Messages during Database Recovery

7 SAPDBA freezes the restored data file so that nobody can access it before recovery has been performed on it.

8 SAPDBA determines which redo log files are needed to recover the data file.

9 SAPDBA asks to restore a newer data file. This is not necessary because we already selected the most recent backup log where the data file is included.

10 SAPDBA starts the recovery process and recovers the data file to its most current state.

10.5.6 Full Offline Database Restore

To restore a SAP R/3 Oracle database to the most current level, SAPDBA calls BRRESTORE to restore:

- The primary copy of the control file
- All of the data files
- The primary copy of the online redo logs
- The parameter files if necessary

While the database is in offline mode, follow these steps:

1. Log in as orac21.
2. Create the necessary control and online redo log file directories.

   SAPDBA generates all directories of the files that the BRBACKUP command saves. But be aware that SAPDBA does not save all files of the SAP R/3 Oracle database. SAPDBA does not save the following files because they are only mirror copies:

   ```
   /oracle/C21/sapdata1/cntrl/ctrlC21.ctl
   /oracle/C21/sapdata2/cntrl/ctrlC21.ctl
   ```

SAPDBA also does not save the mirror copies of the online redo log files located in these directories:

```
/oracle/C21/mirrlogA
/oracle/C21/mirrlogB
```

We recommend that you re-create these directories before you start the restore process, unless they already exist. If you do not re-create these directories before you start the restore process, SAPDBA restores only the primary copy of the control file to /oracle/C21/dbs/ctrlC21.ctl, and you then have to copy the primary control file to the /oracle/C21/sapdata1/cntrl and /oracle/C21/sapdata2/cntrl directories. Alternatively, you can restore the primary control file, using the ADSM client GUI, to the other two control file locations. From the AIX command line, issue the following commands as the Oracle orac21 user:

```
mkdir /oracle/C21/sapdata1/cntrl
mkdir /oracle/C21/sapdata2/cntrl
mkdir /oracle/C21/mirrlogA
mkdir /oracle/C21/mirrlogB
```

3. Start SAPDBA.
4. On the SAPDBA menu (Figure 129 on page 282) select option **m** to identify yourself as the expert user and press Enter.
5. Select option **k**, Restore/Recovery.

   The SAPDBA *Restore / recovery* menu is displayed (Figure 146 on page 305).
6. Select option **a**, Full restore (incl. redo logs and control files). The SAPDBA *Full restore* menu is displayed (Figure 155).

```
                              Full restore

                a  -  Restore database and startup open
                      (no recovery possible)
                b  -  Restore database and startup mount
                      (for manual recovery using (backup controlfile))


                q  -  Return


                Please select  ==> b
```

Figure 155. SAPDBA Full Restore Menu

7. Select option **b**, Restore database and startup mount. The SAPDBA submenu to select a backup log file to restore a complete database is displayed (Figure 156 on page 313).

```
****************** SAPDBA - Restore complete database (incl.) **********

   1 - bcwuepki aff 1998-02-25 13.27.36

Please, enter the number of the BRBACKUP run that you want to restore

Selected BRBACKUP run:   action id: bcwuepki
                       function id: aff
```

Figure 156. SAPDBA Submenu to Select a Backup Log File to Restore

Enter the number of the backup log file from which SAPDBA gets the data file identifications for the files to restore from ADSM. SAPDBA chooses the latest, full, offline operation backup log file, which you use if you want to restore the most recent offline version of the database. If you do not want to restore to the most recent offline database version, select another backup log file. Figure 157 shows the first part of the output of the SAPDBA restore process.

Operation: Preparation

```
26.02.98 14:35 --- Making sure that database is in mount or open state ...

Selected BRBACKUP run:   action id: bcwuepki
                       function id: aff

SAPDBA: Shall SAPDBA use ext. backup utility to find and restore files?  1
(y/n) [y] ==> yes

************************ SAPDBA - Find archive files *************************

SAPDBA: Trying to find archive files using the inquire function of ext. backup
        utility ...

              BACKINT/ADSTAR Distributed Storage Manager
⋮
BKI0005I: Start of backint program at: Thu Feb 26 14:36:06 1998.

#BACKUP SAP___9802251808 /oracle/C21/saparch/C21arch1_211.dbf  2
#BACKUP SAP___9802261431 /oracle/C21/saparch/C21arch1_212.dbf
⋮
SAPDBA: Backup from   1998-02-25 13.27.36
        Recover until now  3
        First log     211
        Last log      213

********************** SAPDBA: Prepare for full restore **********************

SAPDBA: This is the last chance to cancel the restore process of all
        tablespaces of your database. If you continue now all your tablespace
        files will be deleted and overwritten by an older generation. A
        recovery up to now will be applied to your database after restore. The
        database will be opened. If the restore or recovery process fails for
        some reason your database will probably be in an inconsistent or
        incomplete state.

Do you want to continue? (y/n) ·n“ ==> yes

SAPDBA: Backing up the online redo logs ...
SAPDBA: Checking whether BRBACKUP is running ...
SAPDBA: Checking whether R/3 is running ...
SAPDBA: ALTER all TABLESPACEs END BACKUP ...
```

Figure 157. SAPDBA Restore Log File for Full Restore

1 In the first step of the restore process, SAPDBA starts to find the files that are to be restored.

2 SAPDBA starts to find the archive redo log files that possibly are needed for recovery.

3 SAPDBA specifies the point in time for recovery.

SAPDBA performs a shutdown of the database (if necessary) and executes the BRRESTORE command to restore the files from ADSM, using BACKINT/ADSM. Figure 158 shows the next part of the SAPDBA restore process.

Operation: Data File Restore

```
SAPDBA: Instance C21 successfully shutdown.
        Instance C21 shut down.
⋮
List of restore parameters
                              Current value

BRBACKUP profile              initC21.sap
Backup utility parameter file /oracle/C21/dbs/initC21.utl
Language                      English

SAPDBA: Restoring the file(s) of all tablespaces from the backup that was
        created on 1998-02-25 at 13.27.36 (bcwuepki.aff) ... 4

BR401I BRRESTORE 3.0F
BR405I Start of database restore: rcwujtpu rsb  1998-02-26 14.39.42

BR407I Restore of database: C21
BR408I BRRESTORE action ID: rcwujtpu
BR409I BRRESTORE function ID: rsb
BR410I Tablespaces for restore: ALL
BR419I Files will be restored from backup: bcwuepki.aff 1998-02-25 13.27.36
BR416I 27 files found to restore, total size 6589.211 MB.
BR421I Backup device type for restore: util_file
BR256I Please enter 'cont' to continue, 'stop' to cancel the program:
BR257I Your reply: 'cont'
BR259I Program execution will be continued...

BR280I Time stamp 1998-02-26 14.39.49
BR229I Calling backup utility...
BR278I Command output of '/usr/sap/C21/SYS/exe/run/backint -u C21 -f restore
     ... -i /oracle/C21/sapbackup/.rcwujtpu.lst -t file -p /oracle/C21/dbs/initC21.utl':
⋮
BKI0005I: Start of backint program at: Thu Feb 26 14:39:49 1998. 5

BKI0014I: 27 Objects to restore.
BKI0009I: Number of bytes to restore: '6.435 GB'.
BKI0030I: Time: 14:40:08 Object: 1 of 27 in process: /oracle/C21/sapdata4/el30fd_1/el30fd.data1
     ... Size: 498.008 MB. ADSM server: TERASERV.
BKI0030I: Time: 14:40:08 Object: 2 of 27 in process: /oracle/C21/sapdata6/es30fd_1/es30fd.data1
     ... Size: 1492.008 MB. ADSM server: TERASERV.
BKI0030I: Time: 14:40:08 Object: 3 of 27 in process: /oracle/C21/sapdata2/es30fi_1/es30fi.data1
     ... Size: 799.008 MB. ADSM server: TERASERV.
⋮
BR280I Time stamp 1998-02-26 15.29.12
#FILE..... /oracle/C21/sapdata2/user1i_1/user1i.data1
#RESTORED. SAP___9802251327
BKI0054I: Time: 15:29:12 Object: 27 of 27 done: /oracle/C21/sapdata2/user1i_1/user1i.data1
     ... with 2056.000 KB restored with description 'SAP___9802251327'
BKI0049I: Object /oracle/C21/sapdata2/user1i_1/user1i.data1 with 2056.000 KB restored with description
     ... 'SAP___9802251327_TERASERV@R3_DB#011 0002'
```

Figure 158 (Part 1 of 2). SAPDBA Restore Log File for Full Restore

```
BKI0023I: Time: 15:29:12 Done: 6.435 GB (100.0%) of 6.435 GB. Estimated end time:
15:29:12.
BKI0012I: Number of bytes left to be restored : 0 Bytes (0.0%) of 6.435 GB.
⋮
BR374I 27 of 27 files restored by backup utility.
BR230I Backup utility called successfully.

BR406I End of database restore: rcwujtpu rsb  1998-02-26 15.29.12 6
BR402I BRRESTORE terminated successfully.
```

Figure 158 (Part 2 of 2). SAPDBA Restore Log File for Full Restore

4 On the basis of the specified backup log file, BRRESTORE restores all data files of all tablespaces.

5 BACKINT/ADSM starts the restore process and restores all files from ADSM.

6 BRRESTORE successfully restores all data files of all tablespaces.

SAPDBA now creates the mirror copies of the control files and online redo log files as shown in Figure 159.

```
BR351I Restoring /oracle/C21/mirrlogA/log_g11m2.dbf
BR355I from /oracle/C21/origlogA/log_g11m1.dbf ...

BR351I Restoring /oracle/C21/mirrlogB/log_g12m2.dbf
BR355I from /oracle/C21/origlogB/log_g12m1.dbf ...

BR351I Restoring /oracle/C21/mirrlogA/log_g13m2.dbf
BR355I from /oracle/C21/origlogA/log_g13m1.dbf ...

BR351I Restoring /oracle/C21/mirrlogB/log_g14m2.dbf
BR355I from /oracle/C21/origlogB/log_g14m1.dbf ...

BR351I Restoring /oracle/C21/sapdata2/cntrl/cntrlC21.dbf
BR355I from /oracle/C21/sapdata1/cntrl/cntrlC21.dbf ...

BR351I Restoring /oracle/C21/dbs/cntrlC21.dbf
BR355I from /oracle/C21/sapdata1/cntrl/cntrlC21.dbf ...
```

Figure 159. SAPDBA Creates the Mirror Copies

SAPDBA now starts the database in mount mode and performs media recovery. Figure 160 shows the last part of the SAPDBA restore process.

```
Operation: Recovery
********************** SAPDBA - Startup instance C21 ************************
⋮
SAPDBA: Instance C21 mounted.
⋮
********************** SAPDBA - Restore archive files ***********************

SAPDBA: The log sequence of the first archive file to be restored is 211.
SAPDBA: The log sequence of the last  archive file to be restored is 213.
SAPDBA: All archive files from sequence number 211 to 213 found.
SAPDBA: Size of redo log files / archive files: 20480 KB
⋮
```

Figure 160 (Part 1 of 2). SAPDBA Restore Log File for Full Restore

```
SAPDBA: The restore of the archive logs starting at log sequence number 211 to
        213 terminated successfully. 7

****************** SAPDBA - Point in time recover database ******************

SAPDBA: Recovery from log 211 to log 213 (all in one step)  8

SAPDBA: SQLDBA script recov002.sql has been created in directory
        "/oracle/C21/sapreorg/9802261435" for recovery using the restored log
        files.
        For this recovery no script recov001.sql will be needed.

Do you want to start recovery?
Please select (y/n) ·y" ==> yes

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

SAPDBA: Processing SQLDBA command file
        '/oracle/C21/sapreorg/9802261435/recov002.sql':

CONNECT INTERNAL
SET     AUTORECOVERY OFF
RECOVER DATABASE;  9
/oracle/C21/origlogA/log_g13m1.dbf
/oracle/C21/origlogB/log_g14m1.dbf
/oracle/C21/origlogA/log_g11m1.dbf

Oracle Server Manager Release 2.1.4.0.0 - Production

Copyright (c) Oracle Corporation 1994, 1995. All rights reserved.

Oracle7 Server Release 7.2.3.0.0 - Production Release
PL/SQL Release 2.2.3.0.0 - Production

SVRMGR> Connected.
SVRMGR> Autorecovery            OFF
SVRMGR> Media recovery complete.
SAPDBA: Database mounted and opened.

SAPDBA: Database open.

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* SAPDBA : Recover database terminated successfully.  *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

Figure 160 (Part 2 of 2). SAPDBA Restore Log File for Full Restore

7 BRRESTORE restores the archive redo logs to disk.

8 SAPDBA creates a recovery script for applying the archive redo logs. The script is called */oracle/C21/sapreorg/9802261435/recov0002.sql.*

9 SAPDBA recovers the database to the most current point in time, using the previously created recovery script, recov002.sql.

10.5.7 Database Point-in-Time Restore

In situations where logical corruption of the database has been introduced, it might be desirable to restore the database to a specific point in time before the problem occurs.

The following steps are involved in restoring the database to a point in time:

1. Ensure that you have a full backup of an entire SAP R/3 system, including the SAP R/3 Oracle database and all non-database-related files as the starting point.
2. Define a point in time to which you want to recover the database. You have to know the exact date and time in the following format:

YYYY-MM-DD HH:MM:SS

3. Determine the appropriate backup log file that reflects the version of the data files you want to restore (it might not be the most recent version).

   To determine the appropriate backup log file in advance, use the backup log history file, /oracle/C21/sapbackup/backC21.log, and select the backup log file that reflects the date and time of the last full backup of your database. Later, during the restore operation, SAPDBA prompts you to specify the backup log file.

4. Shut down the database:

```
sapdba -shutdown
```

5. Start SAPDBA.

6. Select option **m** from the SAPDBA menu (Figure 129 on page 282) to identify yourself as the expert user and press Enter.

7. Select option **k**, Restore/Recovery. The SAPDBA Restore / recovery menu is displayed (Figure 146 on page 305).

8. Select option **b**, Full restore and recovery (excl. redo logs). The SAPDBA Full Restore and Recovery menu is displayed (Figure 161).

```
                        Full Restore and Recovery

        DATABASE STATE   : Database open
        RESTORE / RECOVER: disallowed (see status)

                                                Current setting

        A - Select a full online/offline backup
        b - Recover until                       now
        c - Show status
        D - Restore and recover

        q - Return

        Please select  ==> A
```

Figure 161. SAPDBA Full Restore and Recovery Menu

9. Choose option **A**, Select a full online/offline backup.

10. SAPDBA asks you to specify the external backup utility and the utility file from which to find and restore the files (not shown). Figure 162 shows the available SAPDBA backup log files.

```
   1 - bcwuepki aff 1998-02-25 13.27.36

Please, enter the number of the BRBACKUP run that you want to restore
(q = quit) [1] ==> 1
```

Figure 162. SAPDBA Backup Log Files

11. Select option **b**, Recover until, on the SAPDBA Full Restore and Recovery menu to enter a date to recover until in the following format: *YYYY-MM-DD HH.MM.SS*. Figure 163 on page 318 shows the updated SAPDBA Full Restore and Recovery menu.

```
                        Full Restore and Recovery

        DATABASE STATE   : Database mounted
        RESTORE / RECOVER: disallowed (see status)

                                                 Current setting

        A - Select a full online/offline backup  1998-02-25 13.27.36
        b - Recover until                        1998-02-25 21.30.00
        c - Show status
        D - Restore and recover

        q - Return

        Please select  ==> D
```

Figure 163. Updated SAPDBA Full Restore and Recovery Menu

12. Select option **D**, Restore and recover, to start the restore and recovery operation. SAPDBA checks all control, online redo log, and data files.
    .

    Note: You must allow SAPDBA to remove these files because you want to restore a different version of the database.

    SAPDBA performs the following activities when restoring a full backup of the database and recovering to a specified point in time:

    a. Backs up the primary control file to the */oracle/C21/sapreorg* directory
    b. Backs up the primary online redo log files to the */oracle/C21/sapreorg* directory
    c. Optionally, deletes all data files
    d. Shuts down the database with the *Shutdown immediate* option
    e. Calls BACKINT/ADSM to restore all of the data files
    f. Starts the database in mount mode

13. BRRESTORE calls the BACKINT/ADSM utility to restore all data files of the Oracle database (not shown here). SAPDBA finds and restores the archived redo logs if necessary (not shown here). Figure 164 on page 319 shows the SAPDBA messages when checking which files need recovery.

```
****** SAPDBA - Checking data file's RECOVERY status via V$RECOVER_FILE ******

SAPDBA: Data file '/oracle/C21/sapdata1/system_1/system.data1' 1
        needs recovery from 0 to 213 for thread 1
        but no restore

Do you rather want to restore a newer data file? (y/n) [n] ==> no 2

⋮

SAPDBA: Data file '/oracle/C21/sapdata6/es30fd_1/es30fd.data1'
        needs recovery from 0 to 213 for thread 1
        but no restore

Do you want to skip this dialog for all further existing datafiles (Always recovery)?
(y/n) [n] ==> no

********************** SAPDBA - SUMMARY OF CHECK RESULTS **********************

SAPDBA: 0 data file(s) need to be restored and recovered 3
        27 data file(s) need to be recovered only


SAPDBA: Database mounted
```

Figure 164. SAPDBA Recovery Messages for Individual Data Files

1 SAPDBA starts checking the recovery status of each data file.

Because the restore process (not shown here) has completed successfully, only recovery for each data file is needed to roll forward the entire database to the specified point in time.

2 SAPDBA asks whether you want to restore a newer file because it knows that more recent backup log files exist. We want to recover to a specific point in time, so we do not want to restore a newer version of the file.

3 SAPDBA summarizes the results of the check operation and reports that the data files need recovery only.

SAPDBA creates a recovery script to recover the database to the specified point in time. The recovery script contains the following Oracle commands to recover the database:

```
CONNECT INTERNAL
SET     AUTORECOVERY OFF
RECOVER DATABASE UNTIL TIME '1998-02-25:21:30:00';
```

SAPDBA executes the script to recover the database to the specified point in time.

14. At the beginning of the recover database operation, SAPDBA prompts you to start the recovery. Enter **yes** to start the recovery process. Figure 165 on page 320 shows the SAPDBA messages for recovering the data files.

```
************************* SAPDBA - Recover database *************************

SAPDBA: Recovery requires online redo logs only  4
        beginning from 213

SAPDBA: Database successfully recovered using online redo logs.

26.02.98 17:59 --- Trying to startup database from mount to open ...

26.02.98 17:59 --- Database mounted and opened.

SAPDBA: Database open

* * * * * * * * * * * * * * * * * * * * * * * * * * * *
* SAPDBA: Recover database terminated successfully. *
* * * * * * * * * * * * * * * * * * * * * * * * * * * *

SAPDBA: End of recovery log file.
        on 26.02.98 at 17:59.
```

Figure 165. SAPDBA Messages for a Point-In-Time Recovery

4 In this example SAPDBA requires only the contents of the online redo logs to recover the database to the specified point in time.

In most cases archive redo logs are necessary to recover the database, especially if the specified point in time is older than the backup version of the data files. In our example we specified a point in time where no archive logs were needed.

10.6 Additional Backup Considerations

This section covers backing up non-database SAP R/3 files and automating backups.

10.6.1 Backup and Restore of SAP R/3 Nondatabase Files

SAP R/3 changes and creates many files outside the database. You have to back up and restore the following files:

- Oracle executables and profiles
- SAP R/3 executables and profiles
- SAP R/3 reorg files needed for reorganization of the database
- Database backup and archive log files

You must save at least the backup and archive log files, which you cannot reinstall from product sources, such as tape or CD-ROM.

BRBACKUP and BRARCHIVE generate and update specific backup and archive log files that you need to properly restore and retrieve database tablespaces. For example, the BRBACKUP command tracks all of its database backup activities in the */oracle/C21/sapbackup/backC21.log* file. This file contains information about the backup log files, which SAPDBA uses to identify the files to restore. The BRARCHIVE command tracks all of its archived redo log activities in the */oracle/C21/saparch/archC21.log* file. The *archC21.log* file contains information about the archive log files, which SAPDBA needs to have for restore and recovery.

During a restore and recovery operation, the backup and archive log files must exist. If, for some reason, the files do not exist when you start the restore operation, SAPDBA does the following:

1. Determines the contents of the backup log files, calling the BACKINT/ADSM utility by using the SAP R/3 INQUIRY function. The INQUIRY function enables queries to the ADSM server for backup IDs and file names of backups currently stored within the ADSM server.

   You can perform the same functions with the BACKINT/ADSM GUI, *backfm*.

2. Starts the restore and recovery process based on this information
3. Determines the backup IDs and versions of data files stored on an ADSM server by looking for the data files with the description IDs (backup IDs).

SAPDBA cannot determine which archived redo log files must be restored and applied for database consistency. You must specify this information when you recover the database with SAPDBA and do not have the backup and archive log files available.

To back up the SAP R/3-related non database files, use either the ADSM dsmc selective or archive command. To keep files together during backup and restore operations, use the ADSM dsmc archive command with the -desc option. This makes the retrieve operation of all files easier because you can use a description to select the appropriate files.

Generally, the ADSM dsmc retrieve command has the disadvantage that it does not re-create symbolic links on the AIX operating system level. A large number of SAP R/3 non database files have symbolic links to other directories. However, the database data files do not have symbolic links. We recommend that you use the ADSM dsmc selective or incremental command to save SAP R/3 non database files and the ADSM dsmc restore command to re-create symbolic links properly during the restore. However, if you have empty directories in your AIX file system structure and you use them as mount points for NFS directories, we strongly recommend that you not use the ADSM dsmc selective command. Instead use the ADSM dsmc incremental command. The ADSM dsmc incremental command backs up empty directories, whereas the ADSM dsmc selective command does not.

The example below shows how to use the ADSM dsmc incremental commands to back up all SAP R/3-related non database files. You can use either modified or absolute modes with the incremental backup. The mode you use depends on your hardware environment. If you want to back up only the modified files, after a specific time period, the files you back up will be distributed over several tape volumes. You then need to run collocation on the tape volumes. If you want to back up all files belonging to an AIX file system, whether or not they have been modified, you can use an absolute backup. In this case, you always have a full backup of the AIX files that are not distributed over several tape volumes. Modified and absolute modes are set with the mode parameter in the copygroup definitions of the management class you use for your backups.

Note: Do not use the ADSM dsmc incremental command for the AIX /oracle/C21/saparch file system, which is where the archived redo logs of the database are located. You do not have to back up the entire file system, just the SAP R/3 log files of the archived redo log backup operation. These files begin with an **a**, and you use the ADSM dsmc selective command with a wildcard to back them up.

For a description of the AIX file systems and directories, see Figure 132 on page 289. Here are the ADSM incremental commands to back up all SAP R/3-related non database files:

```
dsmc incr -node=C21 /oracle/C21
dsmc incr -node=C21 /oracle/C21/sapreorg
dsmc incr -node=C21 /sapmnt/C21
dsmc incr -node=C21 /usr/sap/C21
dsmc incr -node=C21 /oracle/stage
dsmc sel  -node=C21 /oracle/C21/saparch/a*"
```

Here are the corresponding ADSM dsmc restore commands to restore all SAP R/3-related non database files:

```
dsmc rest -node=C21 -subdir=yes  /oracle/C21
dsmc rest -node=C21 -subdir=yes  /oracle/C21/sapreorg
dsmc rest -node=C21 -subdir=yes  /sapmnt/C21
dsmc rest -node=C21 -subdir=yes  /usr/sap/C21
dsmc rest -node=C21 -subdir=yes  /oracle/stage
dsmc rest -node=C21 -subdir=yes  /oracle/C21/saparch/a*"
```

Enhancements

BRBACKUP and BRARCHIVE Version 3.0 can back up the control files as well as the database and redo logs. Version 2.2x provides this enhancement only if you are backing up directly to tape. The files that are backed up are:

```
BRBACKUP
 /oracle/C21/sapbackup/bctkacqx.anf (file name is specific to backup)
 /oracle/C21/sapreorg/structC21.log
 /oracle/C21/sapbackup/backC21.log
 /oracle/C21/sapreorg/reorgC21.log
 /oracle/C21/dbs/initC21.sap
 /oracle/C21/dbs/initC21.utl
 /oracle/C21/dbs/initC21.ora
 /oracle/C21/dbs/initC21.dba

BRARCHIVE
 /oracle/C21/dbs/initC21.sap
 /oracle/C21/saparch/archC21.log
 /oracle/C21/sapreorg/structC21.log
 /oracle/C21/dbs/initC21.utl
 /oracle/C21/dbs/initC21.ora
 /oracle/C21/sapreorg/reorgC21.log
 /oracle/C21/saparch/actlcaeu.svd (file name is specific to archive)
```

Note: BRBACKUP and BRARCHIVE save all parameter files and backup log files that are needed for restore/recovery. The preferred way to restore these files is to use the SAP R/3 and BACKINT/ADSM tools. However, we recommend saving the parameter and backup log files with ADSM commands as well. The reason for this is the fact that the important files will be saved twice and you have an additional copy of the files. Since the parameter files and backup log files do not occupy too much disk space, the amount of time to save them is usually very short, and with the ADSM dsmc incremental command you only save the files that have been changed since the last backup.

10.6.2 SAP R/3 Backup Automation

Performing unattended backups of an entire SAP R/3 system consists of several steps executed within a shell script. We combine several steps from the previous sections in one shell script.

In many cases the decision on which backup strategy to implement depends on several items, such as the:

- Degree of database activity
- Database downtime
- Time needed to restore the entire database and recover it with redo logs
- Hardware-related limitations, such as LAN throughput and tape drive data transfer rates

For a small database (all tablespaces less than 20 GB in total), it is probably better to perform a full offline or online backup daily. For a large database (all tablespaces larger than 20 GB in total), it is probably better to perform a full offline or online backup weekly.

When you perform a full backup weekly, you must save the archived redo logs as frequently as possible so that you always have another copy of the archived redo logs on different tapes if the first copy fails. This approach is called *incremental backup of an Oracle database*. The archived redo logs are very important for recovering a database to the most recent point in time. If you lose data, for example, on a Friday, and the last full backup was taken last Saturday, you must apply all archived redo logs from Saturday to Friday to bring the database up-to-date.

The SAP R/3-related file system files do not change very often, but we recommend that you back them up daily. The advantage is that you keep all of the SAP R/3-related files together so you have a better overview of what you have to restore if you lose data.

To perform a daily backup of an entire SAP R/3 system, we recommend that you use a shell script similar to that shown in Figure 166 on page 324. This script can be scheduled by using cron or the ADSM client scheduler.

The shell script contains:

- SAP R/3 commands to stop all SAP R/3 instances and the database
- SAP R/3 commands to perform a full offline backup of all tablespaces
- SAP R/3 commands to back up the archived redo logs twice
- ADSM commands to back up all SAP R/3-related files
- SAP R/3 commands to start all SAP R/3 instances and the database

```
# ------------------------------------------------------------------------
# /oracle/C21/sapscripts/backup.csh:
# ------------------------------------------------------------------------
#   Stop SAP R/3 instances
# ------------------------------------------------------------------------
su - c21adm -c stopsap
# ------------------------------------------------------------------------
#   SAP tablespaces
# ------------------------------------------------------------------------
su - orac21 -c "brbackup -c -m all -t offline"
# ------------------------------------------------------------------------
#   archived redo logs as the first copy
# ------------------------------------------------------------------------
su - orac21 -c "brarchive -s"
# ------------------------------------------------------------------------
#   archived redo logs as the second copy and delete
# ------------------------------------------------------------------------
su - orac21 -c "brarchive -scd"
# ------------------------------------------------------------------------
#   SAP related file system files and executables
# ------------------------------------------------------------------------
dsmc selective  -subdir=yes  "/oracle/C21/*"           \
                             "/oracle/C21/saparch/*"   \
                             "/oracle/C21/sapreorg/*"  \
                             "/sapmnt/C21/*"           \
                             "/usr/sap/*"              \
                             "/oracle/stage/*"
# ------------------------------------------------------------------------
#   Start SAP R/3 instances
# ------------------------------------------------------------------------
su -c21adm -c startsap

exit 0
```

Figure 166. Shell Script for Daily Backup of an Entire SAP R/3 System

For additional information about other shell scripts, refer to the *BACKINT/ADSM Program Description and Operations* manual.

Part 3. Informix and Sybase Backup and Recovery

Chapter 11. ADSM and INFORMIX-OnLine

This chapter describes the details for backing up INFORMIX-OnLine databases with ADSM. First we look at INFORMIX-OnLine's DBMS structure so that you can understand which files exist and which files you have to factor in to your backup strategy. Then we look at the INFORMIX-OnLine backup utilities. We finish with specific examples of how to integrate ADSM into your backup strategy, including level-0, level-1, level-2, logical log, and specific dbspaces backups. We also show how to save copies of tables by using INFORMIX-OnLine export utilities. We include examples of how to use the latest INFORMIX-OnLine backup utility, ON-Bar, to back up databases directly to ADSM through ADSM's new X/OPEN API. Please note that when we refer to INFORMIX-OnLine in this book we are referring to the INFORMIX-OnLine database server.

11.1 INFORMIX-OnLine DBMS Structure

As shown in Figure 167, the key components of an INFORMIX-OnLine database that you must consider for backup and recovery are:

- The root dbspace
- Noncritical dbspaces
- Logical logs (critical dbspace)
- Physical logs (which are part of the root dbspace by default)
- Configuration files

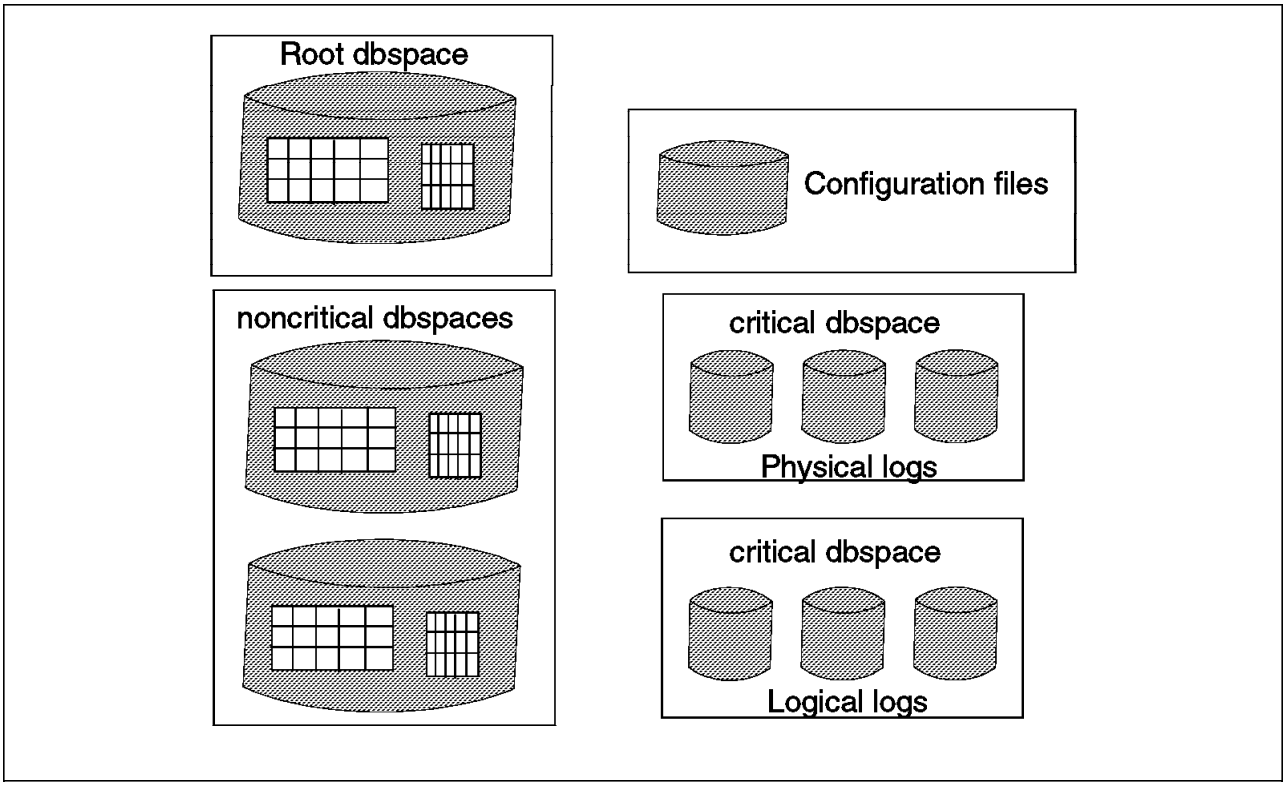


Figure 167. INFORMIX-OnLine DBMS Structure

11.1.1 Root Dbspace

The **root dbspace** is critical because it contains reserved pages and internal tables that describe, control, and track system information. It can also contain the physical and logical logs, and any user databases or tables you want to store in the root dbspace.

There are other critical dbspaces besides the root dbspace, such as any dbspace that contains the physical and logical logs. All other dbspaces are called *noncritical dbspaces*, and they contain all of the user data. This is where the bulk of the "real" database data is held.

Please note that the term *tblspace* for the INFORMIX-OnLine product refers to all of the disk space for a table. This meaning differs from that of *tablespace* in the other RDBMS products. *Dbspace* in INFORMIX-OnLine is comparable in meaning to *table space* in other RDBMS products.

11.1.2 Logical Logs

The **logical logs** contain a history of the database changes. These log files are used to recover a database to *logical consistency*, by rolling forward all committed transactions that have occurred since the checkpoint and rolling back all transactions that were left incomplete. Logical logs also contain changes that are made to the configuration, such as changes to dbspaces and chunks (same meaning as data files in other RDBMSs).

Database logging is the process of writing the transactions to the log files. Different types of logging are supported, such as buffered and unbuffered. It is possible to turn database logging on and off, and to change the type of logging used.

Continuous logical log backups automatically back up each logical log file as it becomes full. It is recommended that you have an offline copy of every log except the current one. Should your system fail then you would only lose the modifications held in the current log.

Once a logical log has been backed up, it can be reused provided that no long transactions are held in it and a checkpoint has been done after the logical log backup. Reusing a logical log minimizes the amount of disk space used for logical logs.

You must use the same utility to back up your logical logs as you use to create your database backups. If you use OnTape to create your dbspace backups, you must use OnTape to back up your logical logs.

11.1.3 Physical Logs

The **physical logs** are used to recover a database to the most recent point of known *physical consistency*, the most recent checkpoint. The physical log contains a "before image" of all physical pages which have been modified since the last checkpoint.

The physical logs are also used to enable a backup to take place with the database online. Such a backup will be a snapshot of the database.

The backup utility forces a checkpoint and then starts to back up database pages, all of which are unmodified. If a page is modified during the backup

process, then an unmodified copy is written to the physical log and the modified copy is marked as “dirty.” When the backup program reaches a dirty page it takes the unmodified copy from the physical log.

Should a checkpoint occur during the backup process, then the checkpoint process causes the backup process to scan the physical log for pages which have not been backed up. Once these pages have been backed up, then the checkpoint completes and the backup program continues to back up unmodified pages until the snapshot is complete.

11.1.4 Configuration Files

The **configuration files** contain all configuration parameters, concerning the root dbspace, disk mirroring, physical and logical logs, message files, tape devices, identification parameters, shared memory parameters, and machine-specific parameters.

For additional explanation of these various database structures, see 2.4, “Fundamentals of Relational Database Management Systems” on page 25. If you run INFORMIX-OnLine on file system files (and plan to use ADSM to directly back up the files, that is, you are not going to use the INFORMIX-OnLine backup utilities at all), we recommend, however, that you:

- Define each dbspace in its own file system to capitalize on the one-to-one relationship between the logical volume and the file system. By doing this, the dbspace will also have a one-to-one relationship with the logical volume. This is discussed in 4.4.2, “RDBMS Design Considerations” on page 63.
- Ensure that the file systems that contain the dbspaces have a mount point on the child directories of your INFORMIX-OnLine directories where the configuration files and executables are located. This eases creation of complete image files of your entire INFORMIX-OnLine environment, including the dbspaces. For example, if you install the INFORMIX-OnLine software in a file system such as /usr/informix5 or /usr/informix6, we recommend that you define your dbspaces in file systems with a mount point on child directories such as /usr/informix5/dbspaces or /usr/informix6/dbspaces. This structure eases ADSM backups when you want to include subdirectories.
- Use the LVM to mirror at least the disk files used for the root dbspace and logical log files and physical logs. You may want to use INFORMIX-OnLine′s internal mirroring instead of LVM to avoid any corruptions that may be placed on both LVM copies.
- Locate the logical log files in their own dbspace and define each dbspace in its own file system. You then have a one-to-one relationship between the logical logs and logical volumes, because file systems and logical volumes have a one-to-one relationship.

11.2 INFORMIX-OnLine Backup Utilities

Several INFORMIX-OnLine utilities can be used for backup purposes. We provide various examples of backup utilities using INFORMIX-OnLine Version 5.0, INFORMIX-OnLine Version 6.0, INFORMIX-OnLine Version 7.1, and INFORMIX-OnLine Version 7.21.

Before we discuss specific backup utilities, it is important to understand a critical difference in terminology between the INFORMIX-OnLine utilities and the storage

manager, ADSM. When data files are sent from INFORMIX-OnLine to ADSM, regardless of the utility used to initiate backup, all data is, by default, stored in backup copy groups. From the perspective of an ADSM administrator, this data is considered “backup” data, not “archive” data.

An **archive** (in INFORMIX-OnLine terminology) backs up all of the data that the database server manages, which includes *all* dbspaces. The archive is written on a tape or to a file and includes all used disk pages. It does not include configuration files, so you have to back them up separately.

Archiving is rather sophisticated in that it provides for true incremental archives so that you do not have to archive the entire database each time. Three levels of archive are provided:

- Level-0, the baseline archive because all used pages are archived
- Level-1, all changes since the last level-0 archive
- Level-2, all changes since the last level-1 archive

You build a schedule for incremental archives that fits the needs of your environment. For example, you can perform level-0 archives monthly, level-1 archives weekly, and level-2 archives daily, or you can build a more frequent schedule. Level-1 and level-2 backups are cumulative since the previous level backup. Given how INFORMIX-OnLine handles incremental backups, we recommend that you carefully consider the ADSM copygroup options, particularly those that determine the number of data versions that exist, retention periods for extra data versions, and copy mode.

The main INFORMIX-OnLine Version 5.0 backup utility is **tbtape**. In INFORMIX-OnLine Version 6.0 tbtape is replaced by **ontape**, which contains enhancements over tbtape. INFORMIX-OnLine Version 6.0 also provides a new backup utility called **ON-Archive** which provides further enhancements over ontape.

We also look at the new INFORMIX-OnLine Version 7.21 and 8.0 utility, **ON-Bar**. Our scenarios document test cases using INFORMIX-OnLine 7.21 only, as INFORMIX-OnLine 8.0 was not available for testing. ON-Bar replaces the functions of ON-Archive and ontape and provides an interface to a storage manager application, such as ADSM.

You can use other INFORMIX-OnLine utilities for backup purposes, but they are actually data migration utilities and not a substitute for tbtape, ontape, ON-Archive, or ON-Bar. The **onunload** (or **tbunload** in INFORMIX-OnLine Version 5.0) and **dbexport** utilities are not coordinated with the information stored in logical log files, and they do not save a copy of system overhead information important to INFORMIX-OnLine.

We show backup examples integrating ADSM with tbtape, dbexport, tbunload, ON-Archive, and ON-Bar, in 11.3, “Using ADSM to Back Up INFORMIX-OnLine” on page 336.

Let us look at these backup utilities in more detail.

11.2.1 Tbtape

Tbtape is the main backup utility for INFORMIX-OnLine Version 5.0 It is used to back up logical logs (including continuous backups), change the database logging status, create and archive, or restore data.

Continuous logical log backups automatically back up each logical log as it becomes full.

Database logging is the process of writing the transactions to the log files. Different types of logging are supported, such as buffered and unbuffered. Tbtape is used to turn database logging on and off, and to change the type of logging used.

A tbtape archive (in INFORMIX-OnLine terminology) backs up all of the data managed by the database server, which includes *all* dbspaces. This is called a *whole system archive*. This archive is written on a tape or to a file and includes all used disk pages. The archive does not include configuration files, so you need to back them up separately.

The tbtape archive is performed when the database is in online or quiescent mode. As discussed in 3.2.2, “Offline Backup” on page 37, quiescent mode can be thought of as an “almost-offline” or single-user administrative mode. The tbtape restore is performed when the database is in offline mode.

11.2.2 Ontape

Ontape is one of the main backup utilities in INFORMIX-OnLine Version 6.0. It is similar to tbtape but provides some significant enhancements. With ontape you can restore specific dbspaces, and the restore can be either warm or cold. Warm restore means that INFORMIX-OnLine can be online, and cold restore means that INFORMIX-OnLine must be offline. Warm restores can be done for noncritical dbspaces, that is, any dbspace other than the root dbspace and the dbspaces that contain the physical and logical logs. Please note that the backup operation is still done in online or quiescent modes and still includes all dbspaces. Only the restore has changed to allow specific dbspaces and online mode.

Another difference between ontape and tbtape is that with ontape you no longer can use the INFORMIX-OnLine onmonitor utility to create the archive. You can only execute the ontape utility from the command line.

11.2.3 ON-Archive

ON-Archive is a powerful new utility for INFORMIX-OnLine Version 6.0. It provides the function of ontape, but adds significant enhancements including:

- Archive of dbspace sets (named collectives of one or more dbspaces)
- Archive and restore of different dbspaces in parallel.

It provides online and quiescent mode backup and online and offline restore for one or more dbspaces in parallel. You group one or more dbspaces into dbspace sets, and ON-Archive works against these dbspace sets.

The output produced by ontape and ON-Archive is incompatible. You cannot back up data with ontape and restore it with ON-Archive (or vice versa).

11.2.4 ON-Bar

ON-Bar (Backup and Restore) is new to the INFORMIX-OnLine product line (in Versions 7.21 and 8.0) and enhances the backup and restore facilities. ON-Bar replaces ON-Archive and ontape and provides the following enhancements:

- Simplifies backup and restore procedures compared to ON-Archive

  Please note that ON-Bar refers to a backup as a backup, not an archive, whereas ON-Archive refers to a backup as an archive.

- Allows for warm backup and restore of dbspaces
- Enables automatic logical log backup through an ALARMPROGRAM variable that calls a customizable script
- Provides automation and management facilities to reduce operator intervention
- Supports SMP, XMP, and distributed data environments.

ON-Bar is designed to support the X/Open Backup Services API (XBSA). Therefore, any storage manager product that supports this standard works with ON-Bar. The term “storage manager” is used here to denote any type of storage management tool, for example, ADSM. An ADSM X/OPEN API is available for AIX and Solaris platforms today. Availability of an ADSM X/OPEN API for an HP-UX platform is planned. Thus ON-Bar backups can be sent directly to ADSM through the ADSM X/OPEN API; no intermediate file is created.

11.2.4.1 Components

Figure 168 on page 333 shows the ON-Bar components: the onbar program, the X/OPEN Backup Services Application Programmer′s Interface (XBSA), the storage manager (ADSM), and the ON-Bar catalog tables, message file, and emergency boot file.

onbar program: The onbar program receives requests to perform backups and restores and passes them to INFORMIX-OnLine or the storage manager. Users can initiate requests either manually, by issuing a command line request to the onbar program from the client machine where the INFORMIX-OnLine database is stored, or automatically, by using the ADSM scheduler to invoke ON-Bar backup or restore. We provide examples of both types of ON-Bar initiation in section 11.3.6, “Full ′Online′ Backup Using ON-Bar and ADSM” on page 373. The ON-Bar utility itself does not provide an independent GUI.

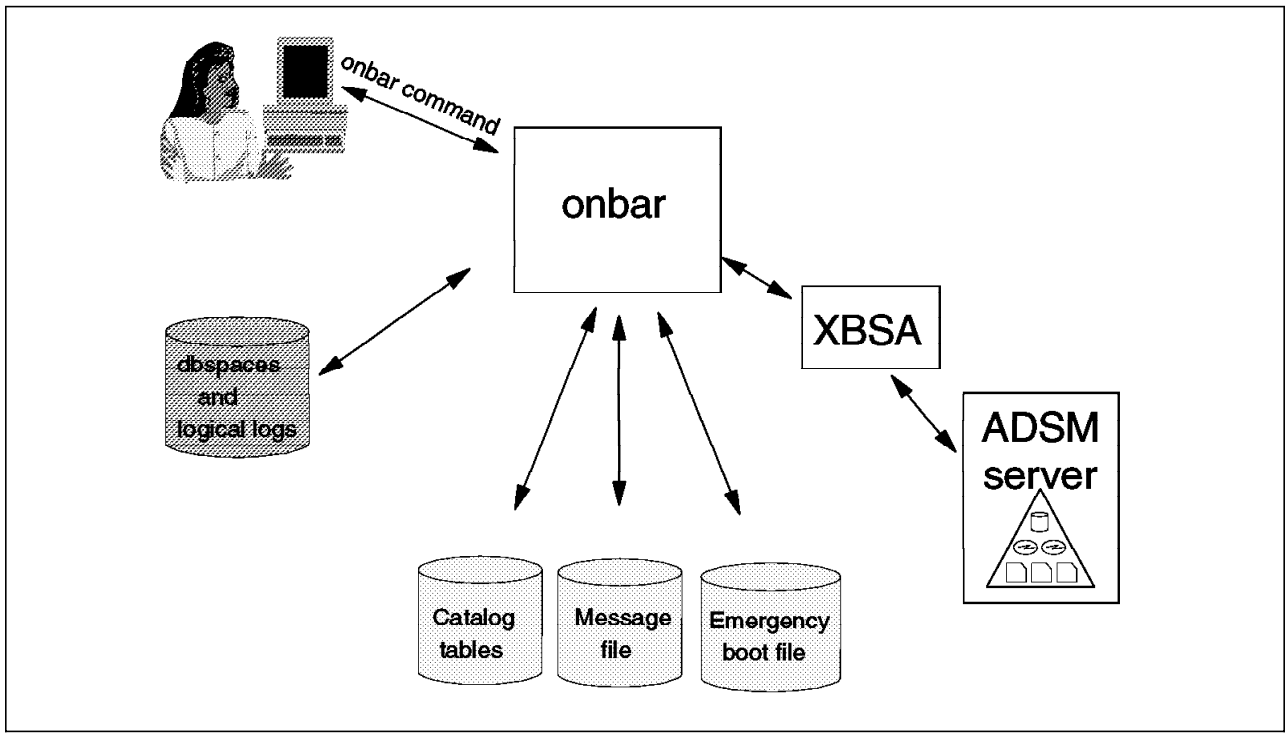


Figure 168. INFORMIX-OnLine ON-Bar Components

Storage Manager: The storage manager handles the actual data storage and media. It may also provide additional capabilities such as the ability to:

- Schedule administrative tasks
- Support a distributed environment
- Enable data compression and decompression

ADSM is a storage manager.

XBSA: ON-Bar and the ADSM storage manager communicate with each other through the XBSA. ON-Bar uses the XBSA to exchange backup and restore data and control data with the ADSM storage manager:

- Backup and restore data

  During backup, ON-Bar receives data copied from INFORMIX-OnLine, such as dbspaces and logs, collects it into objects, and passes the objects to the storage manager through the XBSA. The objects collected on the ADSM storage manager are referred to as *dbobjects* in INFORMIX-OnLine terminology. A *dbobject* may contain a dbspace, blobspace, or logical log file.

  During restore, ON-Bar requests objects from the ADSM storage manager, which retrieves the *dbobjects* and passes them through the XBSA to ON-Bar for restoration on the INFORMIX-OnLine server.

  Logical logs are another type of backup and restore data passed between ON-Bar and the ADSM storage manager. INFORMIX-OnLine allows automatic backup of full logical logs when the ALARMPROGRAM variable in the $INFORMIXDIR/etc/onconfig file references the log_full.sh script. The log_full.sh script automatically invokes a backup of logical logs when they are full by issuing an ON-Bar command. The ON-Bar command in the log_full.sh script passes logical log files through the XBSA interface to the

ADSM storage manager. If you do not want automatic backup of full logs, change the ALARMPROGRAM variable in the $INFORMIXDIR/etc/onconfig file to reference the no_log.sh script. Both the log_full.sh and no_log.sh scripts are located in $INFORMIXDIR/etc.

- Control data

  Control data is passed between ON-Bar and the ADSM storage manager to verify that ON-Bar and the ADSM storage manager X/OPEN API (ADSM's XBSA interface) are compatible, to ensure that objects are restored to the proper instance of the OnLine Dynamic Server in the proper order, and to track the history of backup objects. API code for the ADSM storage manager is packaged with the ADSM client software in the ADSM Application Programming Interface module. ADSM client code at program temporary fix set 2.1.0.5 is the minimum version level required for connectivity with the ON-Bar utility. The ADSM Application Program Interface, Backup/Archive Client, and Common Files modules must be installed on the ON-Bar client node to make the ON-Bar utility interoperate with the ADSM storage manager.

Catalog Tables: The ON-Bar catalog tables, stored in the **sysutils** database, track compatibility of component versions and contain details about *dbobjects*. There are five tables that contain version and backup information:

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

The *INFORMIX-OnLine Dynamic Server Backup and Restore Guide* documents the field contents for each of the above tables in more detail.

Message File: The ON-Bar message file contains information about ON-Bar progress as it executes. The location of this file is determined by the BAR_ACT_LOG variable in the $INFORMIXDIR/etc/$ONCONFIG file. By default, the file is called bar_act.log in the /tmp directory. Review the file to ensure that backups completed successfully on all appropriate database objects, such as database spaces, blobspaces, and log files. The file also provides time stamps on various operations that are useful when you are trying to determine how much time the backups will require to complete so that you can develop ADSM schedules.

Emergency Boot File: The ON-Bar emergency boot file contains enough information to cold restore an online server instance. The boot file is no longer just for critical dbspaces.

The emergency boot file is located in $INFORMIXDIR/etc and is called ixbar.servernum, where servernum is set in the onconfig file. The emergency boot file has two functions. First, it is used, instead of the **sysutils** tables, during a cold restore of the database. During a cold restore, the ON-Bar utility has to request the correct backup objects from the ADSM storage manager. If the

sysutils tables, which usually contain this information, are not available, the data is read from the emergency boot file.

We strongly recommend that you make a backup copy of the emergency boot file each time a dbspace, blobspace, or logical log file backup is performed to ensure some form of manual consistency with information in the ADSM storage manager database. The emergency boot file is a text file that *may*, with the assistance of proper technical support, be modified by hand. We strongly recommend that you take frequent backups of the file because of its critical importance and the difficulty of resolving the contents of the file with the database contents of the ADSM storage manager.

11.2.4.2 Files with Special Backup Requirements

Although the ON-Bar utility is used to back up the INFORMIX-OnLine database objects, several critical files, needed during a recovery period, fall outside the ON-Bar backup strategy. The other files that you must also back up are the:

- $INFORMIXDIR/etc/onconfig file

  Note: Your name for the onconfig file may contain an extension. In our scenarios, we reference an onconfig file named onconfig.learn, where learn is the name of our INFORMIX-OnLine database.

- $INFORMIXDIR/etc/sqlhosts file
- oncfg_$INFORMIXSERVER.SERVERNUM file
- ON-Bar emergency boot file

The contents and purposes of each of the files is documented thoroughly in the INFORMIX-OnLine manuals. As a quick reference, here is a brief description of each file. For examples of the files we used in our environment, see A.3, “INFORMIX-OnLine Version 7.21 Configuration Files” on page 535.

The $INFORMIXDIR/etc/onconfig file is used to tailor characteristics of the overall server instance. It must contain valid parameter settings, or the database will not come up when the oninit command is issued. The $INFORMIXDIR/etc/sqlhosts file is a communication file. It defines how communications are handled between the database and client machines. In our configuration, we configure one communication method for shared memory segments and a second communication method for TCP/IP sockets. The oncfg_$INFORMIXSERVER.SERVERNUM file is used by the INFORMIX-OnLine database server to locate dbspaces. The emergency boot file, $INFORMIXDIR/etc/ixbar.SERVERNUM, records all ON-Bar backup and activity the ON-Bar utility with the ADSM storage manager.

Back up each of these files whenever they are changed. Back up the emergency boot file at least daily and immediately following a backup of a critical dbspace or the root dbspace. You can use the ADSM backup/archive client to back up each file.

11.2.5 Tbunload/tbload (onunload/onload) and dbexport/dbimport

INFORMIX-OnLine provides other utilities that can be used as a supplement (not replacement) for the backup utilities.

The tbunload/tbload (onunload/onload) utility provides the fastest way to move data, but you cannot use it to modify the database schema or migrate from one hardware platform to another. It is the fastest way because it writes in binary in

disk-page units. The database schema (.sql file) contains the database definition statements, such as access privileges and object ownerships. The granularity of data supported is either a table or database.

Dbexport/dbimport provides some flexibility—you can modify the database schema—but you must move the entire database.

For both sets of utilities the database server can be online because the utilities obtain exclusive locks on the tables so that no transaction can update the tables at the time of the commands.

11.2.6 SQL-BackTrack for INFORMIX-OnLine

BMC Software Inc. provides a backup product, SQL-BackTrack, for Oracle, Sybase, and INFORMIX-OnLine. See Chapter 9, “SQL-BackTrack” on page 229 for details on SQL-BackTrack.

11.3 Using ADSM to Back Up INFORMIX-OnLine

In this section we examine how you can use ADSM to back up INFORMIX-OnLine databases either directly or in conjunction with AIX or INFORMIX-OnLine backup utilities. Some examples show databases installed on JFS files; others show databases installed on raw devices. The best alternative is to use ON-Bar because it sends the backup data directly to ADSM through the ADSM X/OPEN API, ADSM′s XBSA interface.

Here are the backup alternatives we performed:

- Full “almost-offline” backup of the database server installed on raw devices, using tbtape and ADSM

  See 3.2.2, “Offline Backup” on page 37 for a description of an “almost-offline” backup.

- Full offline backup of entire INFORMIX-OnLine environment installed on JFS files, using ADSM incremental directly
- Full offline backup of all dbspaces installed on JFS files, using ADSM selective directly
- Full “almost-offline” backup of the database server installed on JFS files, using ontape and ADSM
- Full “almost-offline” backup of the database server installed on JFS files, using ON-Archive and ADSM.
- Full “online” backup of the database server using ON-Bar and the ADSM X/OPEN API
- Partial online backup of two databases (no logs) installed on raw devices, using dbexport and ADSM
- Partial online backup of two tables (no logs) installed on raw devices, using tbunload and ADSM.

Please keep in mind that this is only a subset of possible alternatives. Any examples that use an INFORMIX-OnLine utility first and then ADSM can be implemented for databases installed on either JFS files or raw devices. Either an INFORMIX-OnLine utility, the AIX dd command, or the ADSMPIPE utility must be used first for databases installed on raw devices, because ADSM cannot read

raw devices directly. See 4.2, “Techniques for Using ADSM to Back Up Databases” on page 51 for a general discussion of when you can use ADSM directly and when you must use another utility first.

For an overview of all methods for backing up an INFORMIX-OnLine database, see Figure 379 on page 565.

11.3.1 Full ′Almost-Offline′ Backup (Raw) Using tbtape and ADSM

The tbtape is the main utility to perform a full database server backup for INFORMIX-OnLine Version 5.0. However, the degree to which you can use the utility with ADSM depends on how much free DASD space you a have because tbtape can create a large file, depending on your archive strategy. Using tbtape first and then ADSM is one alternative when your dbspaces are installed on raw devices.

For our example, we implement an archive strategy of weekly level-0 backups and daily level-1 backups. The level-1 backup file grows each day because the data that has changed since the latest level-0 archive accumulates. You can define an ADSM management class that allows seven backup copies. Thus you can recover from a database or disk failure to any day during the week.

The steps we look at are:

- Building a level-0 archive with tbtape
- Using ADSM to back up the level-0 archive
- Building a logical log backup with tbtape
- Using ADSM to back up the logical log backup
- Building a level-1 archive with tbtape
- Using ADSM to back up the level-1 archive
- Restoring the database server from the level-0 archive, level-1 archive, and the logical logs.

11.3.1.1 Building a Level-0 Archive with tbtape

First we need to build a level-0 archive with the INFORMIX-OnLine Version 5.0 tbtape utility. This can be done either interactively using the INFORMIX-OnLine tbmonitor utility or the command line interface of tbtape.

To use the tbmonitor utility issue the following command:

```
tbmonitor
```

Then type a lowercase *a* for the (A)rchive menu or a lowercase *c* for the create (C)reate menu. We enter *a* (not shown) to create a level-0 archive. Figure 169 on page 338 appears, and we enter **0**:

```
Performing archive of the entire INFORMIX-OnLine system.

Please mount tape and press Return to continue ...

Please enter the level of archive to be performed (0, 1, or 2) 0

Archive in progress,  70 percent done.
Please label this tape as number 1 in the archive sequence.

Level 0 archive is 100 percent completed.

date:  Fri May 13 11:16:01 1994

Tapes required to restore the system to the state
at the beginning of this archive:

Archive level:  0     Archive date:  Fri May 13 11:16:01 1994

Logical log unique id at the beginning of the archive:  1

Program over.
```

Figure 169. Using INFORMIX-OnLine tbtape to Build a Level-0 Archive

11.3.1.2 Using ADSM to Back Up the Level-0 Archive

Each tbtape archive writes the output to a /home/informix5/tapedev file. To better separate the level-0 and level-1 archives, we suggest renaming the tapedev file after each backup before you perform the ADSM dsmc selective command to back it up:

- tapedev.0 for a level-0 backup
- tapedev.1 for a level-1 backup
- tapedev.2 for a level-2 backup

The appropriate AIX command to rename the /home/informix5/tapedev file to /home/informix5/tapedev.0 is:

```
mv /home/informix5/tapedev /home/informix5/tapedev.0
```

Figure 170 on page 339 shows the ADSM dsmc selective command to back up the /home/informix5/tapedev.0 file.

```
# dsmc selective -password=mars /home/informix5/tapedev.0

ADSTAR Distributed Storage Manager
Command Line Backup Client Interface - Version 1, Release 2, Level 0.0
(C) Copyright IBM Corporation, 1990, 1994, All Rights Reserved.

Selective Backup function invoked.

Session established with server BALTIC: AIX-RS/6000
 Server Version 1, Release 2, Level 0.0
 Server date/time: 05/13/1994 11:45:20   Last access: 05/13/1994 08:42:37

Normal File-->       9,273,344 /home/informix5/tapedev.0 ..... Sent
Selective Backup processing of '/home/informix5/tapedev.0' finished with no failures.
```

Figure 170. Use ADSM to Back Up the File Created by INFORMIX-OnLine tbtape

After the ADSM backup activity has completed successfully, you have to rename the /home/informix5/tapedev.0 file to its original file name, /home/informix5/tapedev; otherwise INFORMIX-OnLine cannot open the file at the next tbtape operation.

11.3.1.3 Building a Logical Log Backup with tbtape

To properly restore a database server to a consistent state at any point before a disk failure occurred, you must back up the logical logs continuously. The tbtape utility provides a "continuous backup" option that automatically backs up every logical log that becomes full and appends the contents of the logical log to a tape or file.

Usually you back up logical logs to a tape device, but, when using ADSM, you back up the logical logs to a file on disk and then use ADSM to back up the file on disk to an ADSM storage pool. When backing up logical logs to a file system file, in the case of a disk failure, you can also lose the contents of the log file. This is why we recommend mirroring the log file.

To take a continuous backup of the logical logs, use the tbtape utility with the -c option in a dedicated window or screen (Figure 171).

```
$ tbtape -c

Performing continuous backup of logical logs.

Please mount tape and press Return to continue ...
```

Figure 171. INFORMIX-OnLine tbtape Command for Continuous Logical Log Backup

If you want to stop continuous backup of the logical logs, press Ctrl-c to interrupt the tbtape processing. The screen shown in Figure 172 on page 340 displays after an interrupt.

```
Interrupt received ...

Backup of logical log will finish before option is canceled.

Please label this tape as number 1 in the log tape sequence.

This tape contains the following logical logs:

         4 - 5

Program canceled.
$
```

Figure 172. Stopping Continuous Backup of Logical Logs in INFORMIX-OnLine

11.3.1.4 Using ADSM to Back Up the Logical Log Backup

Before you use ADSM to back up the logical log file, you have to rename it so that you can easily restore a sequence of several logical log files. In our example, we rename the created logical log file, /home/informix5/ltapedev, to /home/informix5/ltapedev.1. Use the AIX mv command to rename the /home/informix5/ltapedev file to /home/informix5/ltapedev.1:

```
mv /home/informix5/ltapedev /home/informix5/ltapedev.1:
```

Figure 173 shows the ADSM dsmc selective command to back up the /home/informix5/ltapedev.1 file.

```
# dsmc selective -password=mars -quiet /home/informix5/ltapedev.1

ADSTAR Distributed Storage Manager
Command Line Backup Client Interface - Version 1, Release 2, Level 0.0
(C) Copyright IBM Corporation, 1990, 1994, All Rights Reserved.
```

Figure 173. ADSM Command to Back Up the INFORMIX-OnLine Logical Log

After the ADSM back up has completed successfully, you have to rename the /home/informix5/ltapedev.1 file to its original file name, /home/informix5/ltapedev; otherwise INFORMIX-OnLine cannot open the file at the next logical log back up.

11.3.1.5 Building a Level-1 Archive with tbtape

The next backup performed in our example is a level-1 archive on the following day. Perform a level-1 archive to the /home/informix5/tapdev file.

Use either the tbmonitor utility (to interactively perform the level-1 archive) or the tbtape command line interface. For the level-0 archive we used the tbmonitor utility, so for the level-1 archive let us show you how to use the tbtape command line interface.

Figure 174 on page 341 shows the level-1 archive dialog. Enter **1** for the level of archive to be performed.

```
$ tbtape -s
Performing archive of the entire INFORMIX-OnLine system.

Please mount tape and press Return to continue ...

Please enter the level of archive to be performed (0, 1, or 2) 1
Archive in progress, 0 percent done.
Please label this tape as number 1 in the archive sequence.


Level 1 archive is 100 percent completed.

date:  Sat May 14 11:39:00 1994

Tapes required to restore the system to the state
at the beginning of this archive:

Archive level:  0     Archive date:  Fri May 13 11:16:01 1994
Archive level:  1     Archive date:  Sat May 14 11:39:00 1994

Logical log unique id at the beginning of the archive:  1

Program over.
```

Figure 174. Performing a Level-1 Archive with INFORMIX-OnLine tbtape

11.3.1.6 Using ADSM to Back Up the Level-1 Archive

Figure 175 shows the ADSM dsmc selective command to back up the file that resulted from the level-1 archive, /home/informix5/tapedev.1.

```
# dsmc selective -password=mars /home/informix5/tapedev.1

ADSTAR Distributed Storage Manager
Command Line Backup Client Interface - Version 1, Release 2, Level 0.0
(C) Copyright IBM Corporation, 1990, 1994, All Rights Reserved.

Selective Backup function invoked.

Session established with server BALTIC: AIX-RS/6000
 Server Version 1, Release 2, Level 0.0
 Server date/time: 05/14/1994 11:45:20   Last access: 05/13/1994 11:42:37


Normal File-->       3,686,400 /home/informix5/tapedev.1 ..... Sent
Selective Backup processing of '/home/informix5/tapedev.1' finished with no failures.
```

Figure 175. ADSM Command to Back Up the INFORMIX-OnLine Level-1 Archive

After the ADSM backup completes successfully, you have to rename the /home/informix5/tapedev.1 file to its original file name , /home/informix5/tapedev; otherwise INFORMIX-OnLine cannot open the file at the next tbtape operation.

11.3.1.7 Restoring Level-0 and Level-1 Archives, and Logical Logs

To rebuild your database server, you have to restore the files from ADSM storage and then use tbtape to rebuild your server. As shown in Figure 176, first we use ADSM to restore the latest copy of the level-0 archive file, /home/informix5/tapedev.0, to its INFORMIX-OnLine specific destination file name, /home/informix5/tapedev.

```
# dsmc restore -password=mars -quiet -latest /home/informix5/tapedev.0 /home/informix5/tapedev

ADSTAR Distributed Storage Manager
Command Line Backup Client Interface - Version 1, Release 2, Level 0.0
(C) Copyright IBM Corporation, 1990, 1994, All Rights Reserved.

File /home/informix5/tapedev already exists, do you want to replace it ? (Yes/No) y
```

Figure 176. ADSM Restore of INFORMIX-OnLine Level-0 Archive

Then, as shown in Figure 177, we use tbtape to continue the restore procedure.

```
$ tbtape -r
INFORMIX-OnLine must be off-line to execute the Restore option.
$ tbmode -ky
$ tbtape -r
Restoring entire INFORMIX-OnLine system.
Please mount the first system archive tape and press Return to continue ...
Restoring system control information, please wait ...

                         CHUNKS THAT WILL BE RESTORED

Full Pathname Of The Chunk                                Offset         Size
/dev/rinf5rootdbs                                              0        20000
/dev/rinf5rd                                                   0        20000

Verifying physical disk space, please wait ...
Verifying physical disk space, please wait ...

Would you like to back up log 1? (y/n) n
Restoring dbspaces/BLOBSpaces, please wait ...
Do you have another level of tapes to restore? (y/n)y
```

Figure 177. Continue Restore of INFORMIX-OnLine Level-0 with tbtape

Tbtape prompts you for another level of tapes to restore. We have level-1 archives, so we answer yes to this prompt. To restore the level-1 archives, first we use ADSM to restore the latest copy of the level-1 archive file, /home/informix5/tapedev.1, to the INFORMIX-OnLine specific destination file, /home/informix5/tapedev. We have to perform the restore of the /home/informix5/tapedev.1 file in another screen because we cannot interrupt the tbtape session while rebuilding the database server. Figure 178 on page 343 shows the ADSM restore command.

```
# dsmc restore -password=mars -quiet -latest /home/informix5/tapedev.1 /home/informix5/tapedev

ADSTAR Distributed Storage Manager
Command Line Backup Client Interface - Version 1, Release 2, Level 0.0
(C) Copyright IBM Corporation, 1990, 1994, All Rights Reserved.

File /home/informix5/tapedev already exists, do you want to replace it ? (Yes/No) y
```

Figure 178. ADSM Command to Restore INFORMIX-OnLine Level-1 Archive

After restoring the /home/informix/tapedev.1 file to the INFORMIX-OnLine specific tapedev file, continue with the tbtape process to complete the database server restore activity. Figure 179 shows the second part of the tbtape process to complete the restore of the database server with the level-1 archive.

```
Restoring entire INFORMIX-OnLine system.

Please mount the first system archive tape and press Return to continue ...
Restoring system control information, please wait ...

                        CHUNKS THAT WILL BE RESTORED

Full Pathname Of The Chunk                              Offset         Size
/dev/rinf5rootdbs                                            0        20000
/dev/rinf5rd                                                 0        20000

Verifying physical disk space, please wait ...
Verifying physical disk space, please wait ...
Restoring dbspaces/BLOBSpaces, please wait ...
Do you have another level of tapes to restore? (y/n) n

Initializing, please wait ...

Roll forward should start with log number 1
```

Figure 179. Continue Restore of INFORMIX-OnLine Level-1 Archive with tbtape

After successfully restoring the level-0 and level-1 backup files, restore the logical log files. First use ADSM to restore the logs and then use tbtape to complete the database rebuild. Figure 180 shows the ADSM restore command to restore the logical log file, /home/informix5/ltapedev.1, to /home/informix5/ltapedev.

```
# dsmc restore -password=mars -quiet /home/informix5/ltapedev.1 /home/informix5/ltapedev

ADSTAR Distributed Storage Manager
Command Line Backup Client Interface - Version 1, Release 2, Level 0.0
(C) Copyright IBM Corporation, 1990, 1994, All Rights Reserved.

File /home/informix5/ltapedev already exists, do you want to replace it ?(Yes/No) y
```

Figure 180. Use ADSM to Restore the INFORMIX-OnLine Logical Log

Continue with the tbtape restore process by specifying a logical log tape to restore. Figure 181 on page 344 shows the last part of the entire tbtape rebuild process, the application of the logical logs.

```
Is there  a logical log tape to restore? (y/n) y

Please mount tape and press Return to continue ...
Processing logical logs, please wait ...

Is there another logical log to restore? (y/n) n
$
```

Figure 181. Applying the INFORMIX-OnLine Logical Logs

Now your INFORMIX-OnLine database server should be up and running!

11.3.2 Full Offline Backup (JFS) Using ADSM Incremental Directly

If your INFORMIX-OnLine database is installed on JFS files, you can use ADSM to back up the database directly. You do not need to use an intermediate AIX or INFORMIX-OnLine utility. An advantage of using ADSM directly is that no additional disk space is required to hold the files created by AIX or INFORMIX-OnLine utilities.

This example shows the use of the ADSM incremental utility. When you perform a backup using ADSM directly (without RDBMS commands first), the database server must be offline to ensure consistency between the dbspaces. Please note that it is not necessary to back up any log files because log forward recovery is not an option when the data is recovered from an offline backup.

The ADSM incremental backup is performed from the GUI of the ADSM backup/archive client. Remember to put the MAKESParsfile=no parameter in the options file to correctly handle sparse files (see 4.3.1, “Handling Sparse Files” on page 56 for details). Figure 182 on page 345 shows the ADSM client’s main GUI window. Manager main window. The /usr/informix5 and /usr/informix5/dbspaces files systems are selected for backup.

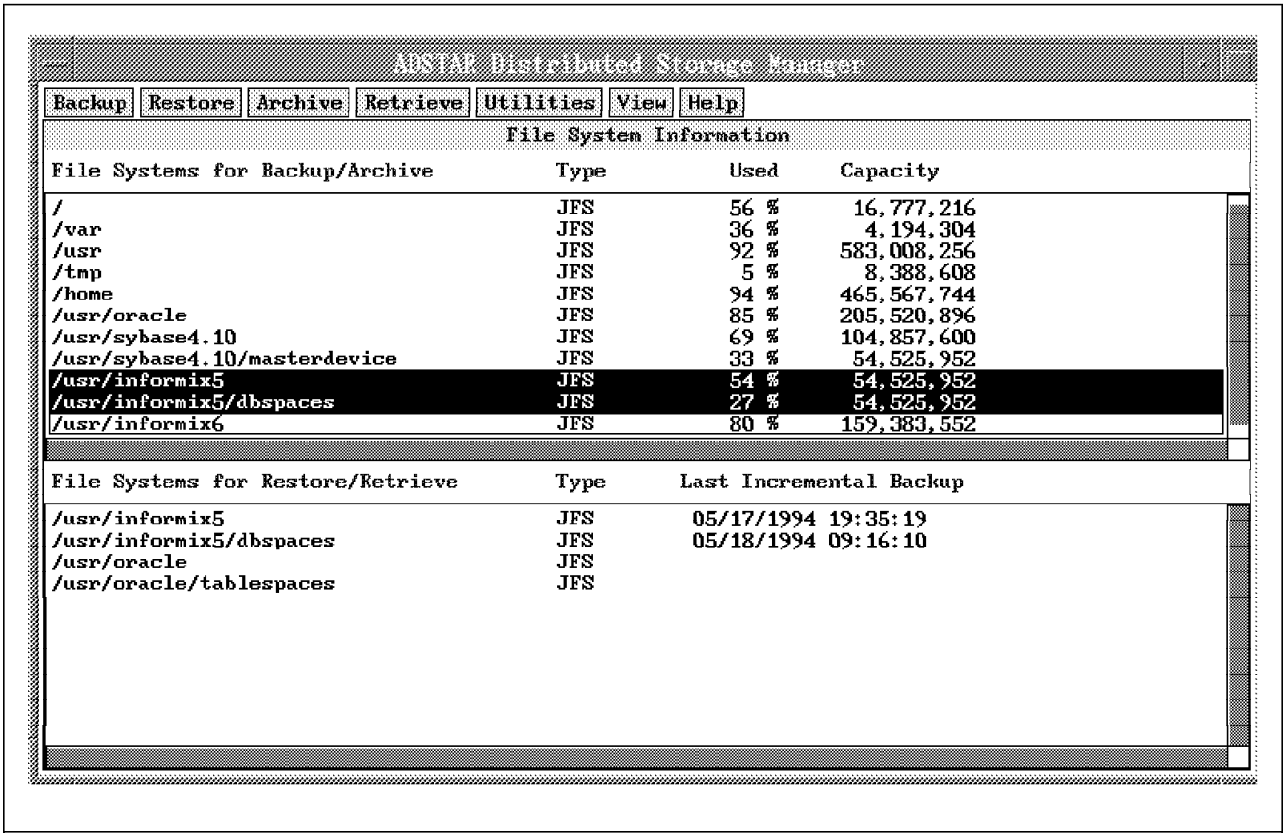


Figure 182. Select All INFORMIX-OnLine Data for Backup

Click on Backup and select Incremental from the pull-down menu (not shown). The Backup Status window appears (Figure 183 on page 346) indicating which files ADSM is backing up. ADSM backs up only those files that have changed since the last ADSM incremental backup. Using incremental backup ensures that if the INFORMIX-OnLine configuration files have changed, they are backed up along with the dbspaces. Thus, at restore time, the contents of the database server and configuration files are consistent. When the backup is complete, an ADSM - Information window is displayed.

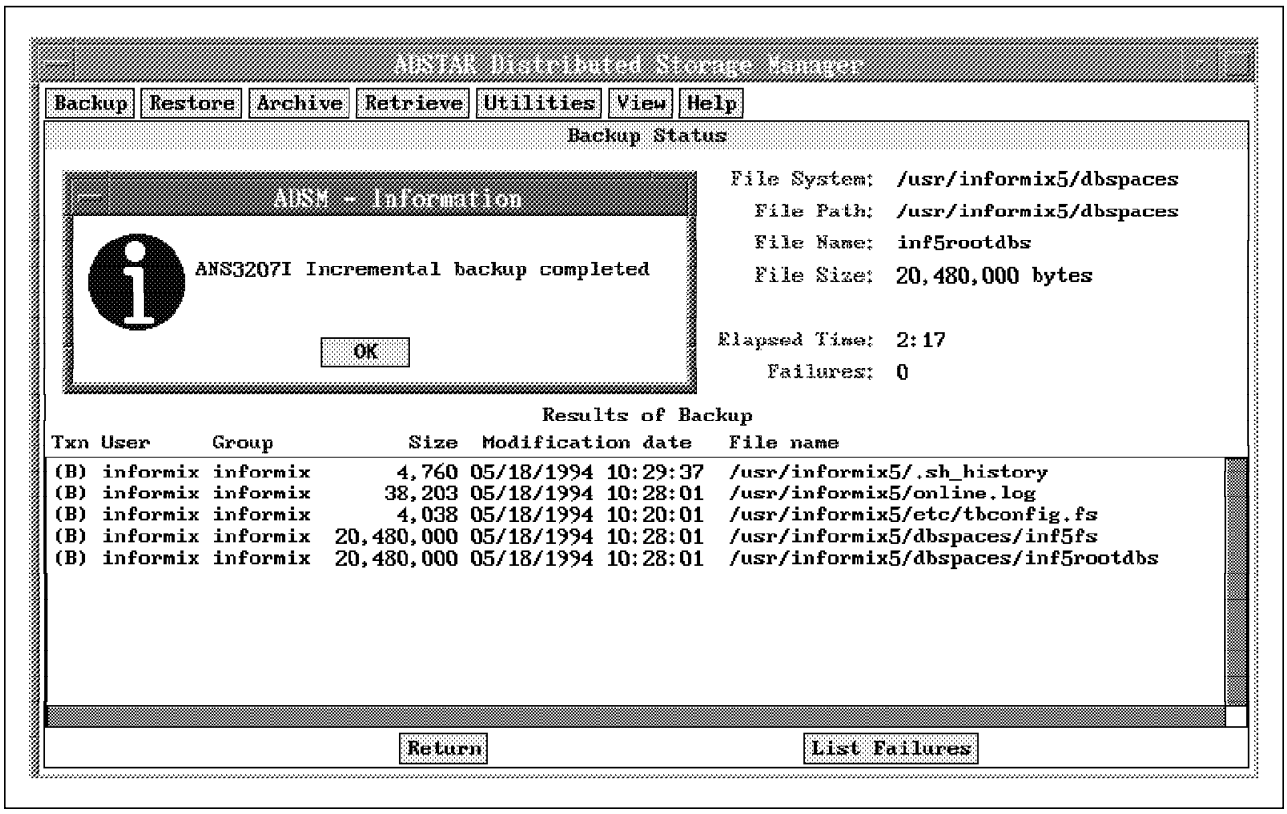


Figure 183. ADSM Backup of All INFORMIX-OnLine Data Is Complete

Let us assume that you have lost your entire INFORMIX-OnLine database server file system: /usr/informix5. The entire database server file system includes the executables, binaries, configuration files, and dbspaces (informix5/dbspaces/inf5rootdbs and informix5/dbspaces/inf5fs files). To restore your database server, you can use ADSM's restore by subdirectory path option. Let us follow this process using the ADSM backup/archive client GUI.

On the main window for the ADSM backup/archive client GUI (Figure 184 on page 347), we select the file systems /usr/informix5 and /usr/informix5/dbspaces to be restored.

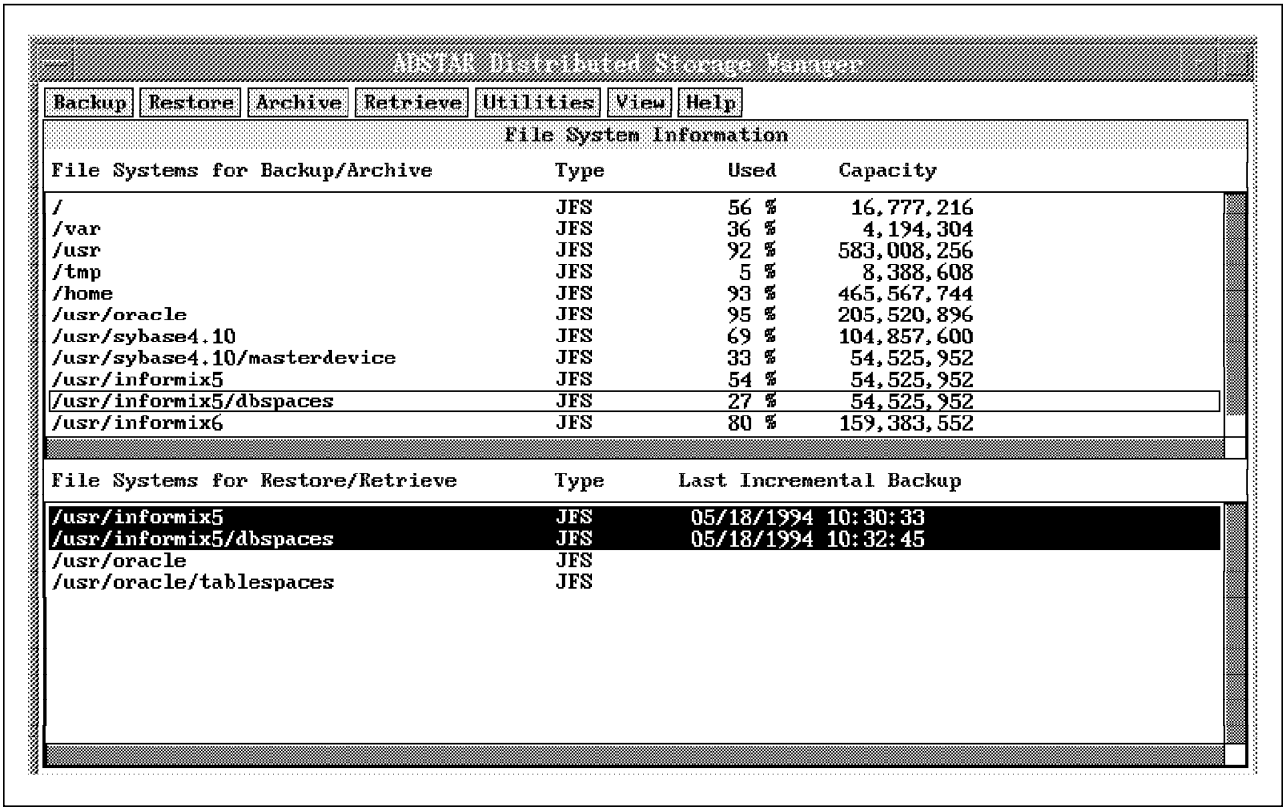


Figure 184. Selecting INFORMIX-OnLine Files for ADSM Restore

Click on Restore and then select the Restore by subdirectory path option (not shown). As shown in Figure 185 on page 348, the ADSM - Restore Subdirectory Path window appears. Enter the source and destination paths and click on OK. This selection restores the entire AIX /usr/informix5 file system and its subdirectories.

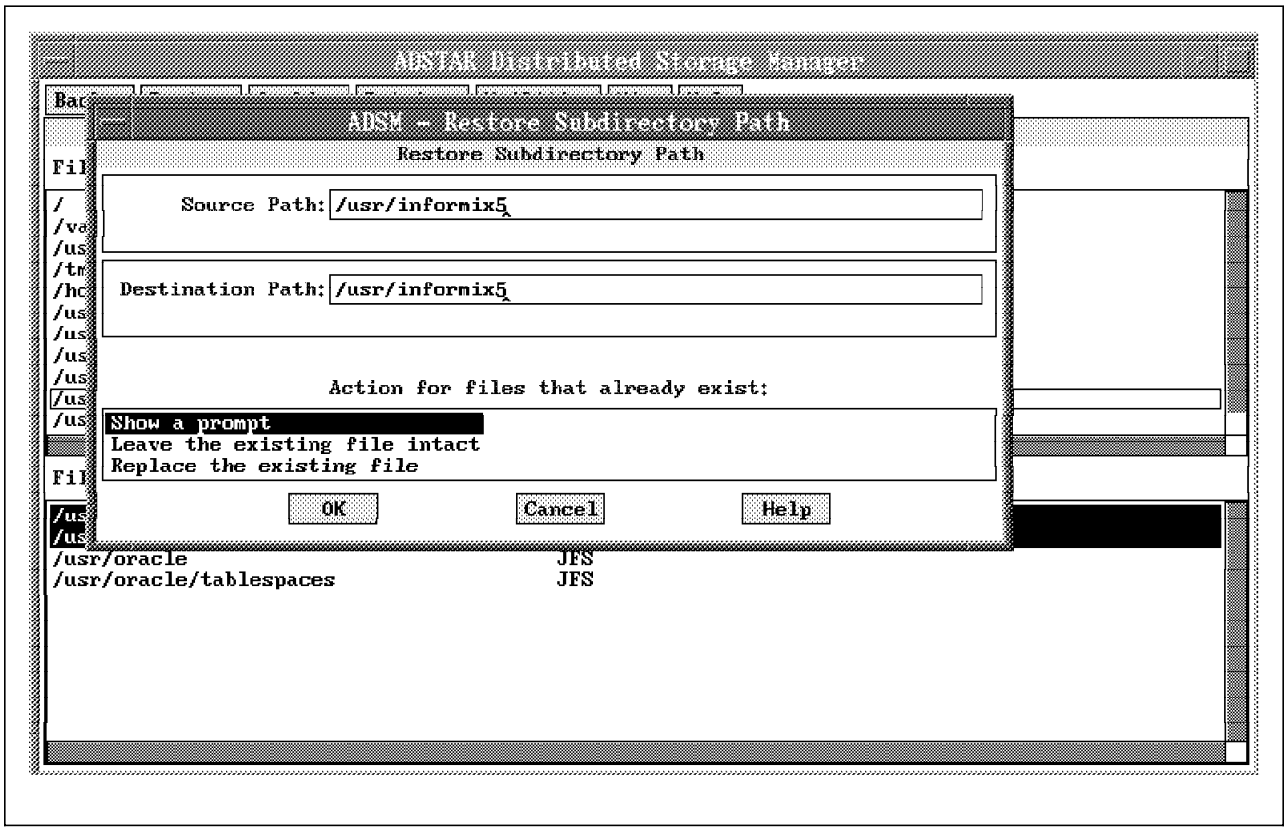


Figure 185. Restore All INFORMIX-OnLine Subdirectories

Figure 186 on page 349 shows the results of the ADSM restore activity for the AIX /usr/informix5 file system.

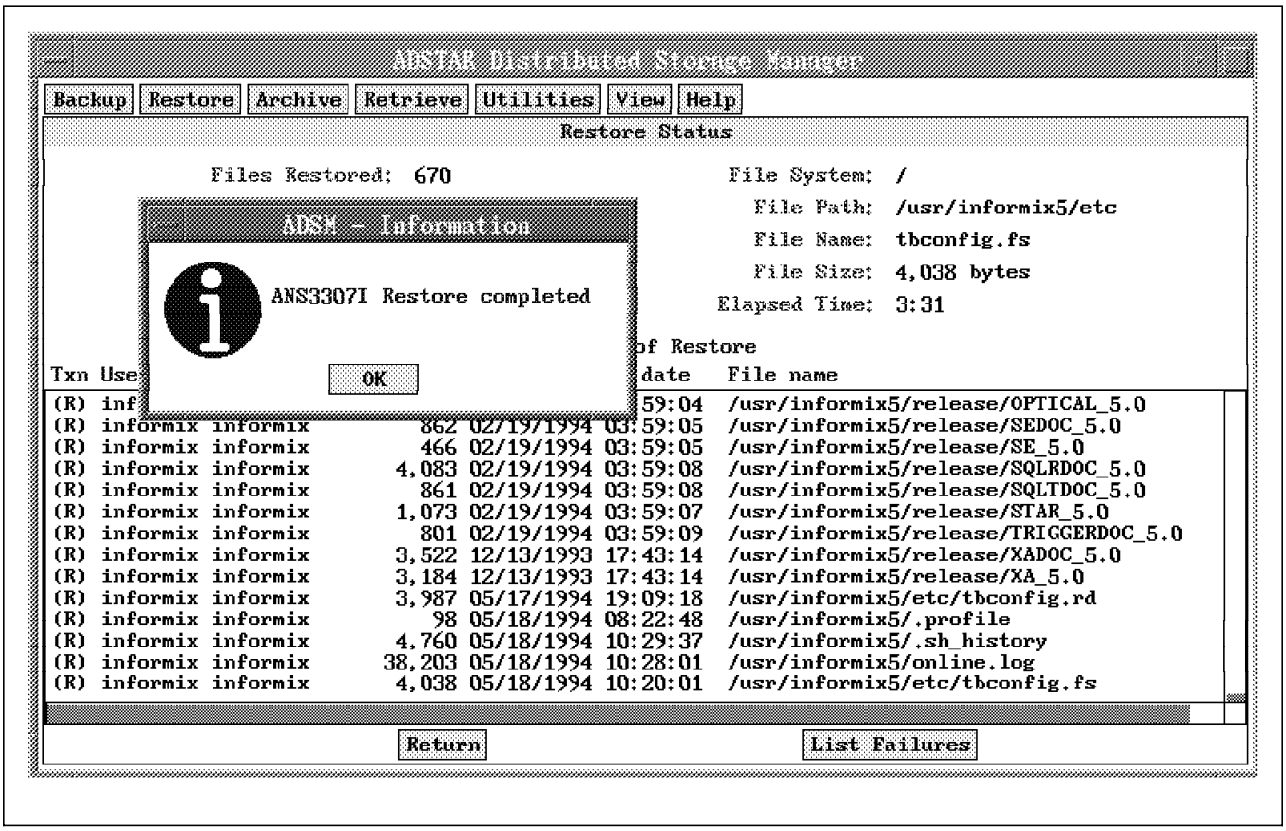


Figure 186. Restore of All INFORMIX-OnLine Data Is Complete

To complete the restore activity, you repeat this process to restore the AIX /usr/informix/dbspaces file system, which contains the INFORMIX-OnLine dbspaces.

11.3.3 Full Offline Backup (JFS) Using ADSM Selective Directly

If your database environment does not change very often, such that the configuration files are not changed, you can back up the INFORMIX-OnLine dbspaces with the ADSM dsmc selective command. ADSM selective provides a command line interface to back up desired files or directories and is useful when you perform an unattended backup through shell scripts that evoke the AIX cron facility.

When you perform a backup using ADSM directly (without RDBMS commands first), the database server must be offline to ensure consistency between the dbspaces. Please note that it is not necessary to back up any log files because log forward recovery is not an option when the data is recovered from an offline backup.

Use the AIX *ls* command to list the file system files that comprise the INFORMIX-OnLine dbspaces.

```
ls -al /usr/informix5/dbspaces
```

Then, as shown in Figure 187 on page 350, enter the ADSM dsmc command to back up the dbspaces including the subdirectories.

```
# dsmc selective -subdir=yes -password=mars /usr/informix5/dbspaces
ADSTAR Distributed Storage Manager
Command Line Backup Client Interface - Version 1, Release 2, Level 0.0
(C) Copyright IBM Corporation, 1990, 1994, All Rights Reserved.

Selective Backup function invoked.

Session established with server BALTIC: AIX-RS/6000
  Server Version 1, Release 2, Level 0.0
  Server date/time: 05/06/1994 14:39:19   Last access: 05/06/1994 14:35:45
Normal File-->      20,480,000 /usr/informix5/dbspaces/inf5fs ......Sent
Normal File-->      20,480,000 /usr/informix5/dbspaces/inf5rootdbs .......Sent
Selective Backup processing of '/usr/informix5/dbspaces' finished with
no failures
```

Figure 187. Enter ADSM Selective Command to Back Up INFORMIX-OnLine

To restore the dbspaces, issue the ADSM dsmc command, remembering to specify the MAKESParsefile=no option (see Figure 188.)

```
# dsmc restore -subdir=yes -makesp=no -pick /usr/informix5/dbspaces
ADSTAR Distributed Storage Manager
Command Line Backup Client Interface - Version 1, Release 2, Level 0.0
(C) Copyright IBM Corporation, 1990, 1994, All Rights Reserved.

Restore function invoked.

Please enter password for node "BALTIC":
ADSM Scrollable PICK Window - Restore

     #    Backup Date/Time  File Size A/I  File
        ---------------------------------------------------------------------
     1. | 05/06/1994 14:47:55    20480000  A   /usr/informix5/dbspaces/inf5fs
     2. | 05/06/1994 14:48:59    20480000  A   /usr/informix5/dbspaces/inf5rootdbs
        |
        |
        |
        0---------10--------20--------30--------40--------50--------60--------7
<U>=Up  <D>=Down  <T>=Top  <B>=Bottom  <R#>=Right  <L#>=Left
<G#>=Goto Line #  <#>=Toggle Entry  <+>=Select All  <->=Deselect All
<#:#+>=Select A Range <#:#->=Deselect A Range  <O>=Ok  <C>=Cancel
pick> +
```

Figure 188. Enter ADSM Restore Command to Restore INFORMIX-OnLine

Because we used the *-pick* option on the restore command, ADSM displays a list of files that match the file specification we entered (/usr/informix5/dbspaces). We selected to restore all of the files displayed by entering a plus (+) sign at the pick command line (pick>). As shown in Figure 189 on page 351, ADSM marks our selections with an **x**. To restore the selected files, enter a lowercase **o** for (O)k on the pick command line. Answer yes to the ADSM prompts for replacing the existing files and the "Restore processing finished." message is displayed.

```
ADSM Scrollable PICK Window - Restore

     #    Backup Date/Time  File Size A/I  File
        ------------------------------------------------------------------------
x    1. | 05/06/1994 14:47:55    20480000  A   /usr/informix5/dbspaces/inf5fs
x    2. | 05/06/1994 14:48:59    20480000  A   /usr/informix5/dbspaces/inf5rootdbs
        |
        |
        |
        0---------10--------20--------30--------40--------50--------60--------7
<U>=Up  <D>=Down  <T>=Top  <B>=Bottom  <R#>=Right  <L#>=Left
<G#>=Goto Line #  <#>=Toggle Entry  <+>=Select All  <->=Deselect All
<#:#+>=Select A Range <#:#->=Deselect A Range  <O>=Ok  <C>=Cancel
pick> o

File /usr/informix5/dbspaces/inf5fs already exists, do you want to replace it?
(Yes/No) y
Restoring   20,480,000 /usr/informix5/dbspaces/inf5fs ..............Done
File /usr/informix5/dbspaces/inf5rootdbs already exists, do you want to replace
 it? (Yes/No) y
Restoring   20,480,000 /usr/informix5/dbspaces/inf5rootdbs .........Done
Restore processing finished.
```

Figure 189. Restore of INFORMIX-OnLine Database Is Complete

11.3.4 Full ′Almost-Offline′ Backup Using Ontape and ADSM

As discussed in 11.2, “INFORMIX-OnLine Backup Utilities” on page 329, the main enhancement in ontape, as compared to tbtape, is the ability to restore specific dbspaces in online mode. The activities of performing a full database server level-0 back up with ontape are very similar to those of tbtape, except you cannot use the onmonitor utility to create the archive; you can only use ontape from the command line.

Because the backup process for ontape is similar to tbtape, we skip the step on how to back up the logical logs. For details on the steps to back up the logical logs continuously, see 11.3.1, “Full ′Almost-Offline′ Backup (Raw) Using tbtape and ADSM” on page 337.

The only differences in this ontape example, compared to the tbtape example, are:

- Use ontape from the command line instead of onmonitor.
- Show ADSM backup from the GUI instead of the command line.
- Perform online (warm) restore.

Figure 190 on page 352 shows the ontape command to perform a level-0 backup of the entire INFORMIX-OnLine Version 6.0 database server.

```
$ ontape -s
Please enter the level of archive to be performed (0, 1, or 2) 0

Please mount tape 1 on /home/informix6/tapedev and press Return to continue ...

100 percent done.

Please label this tape as number 1 in the arc tape sequence.
This tape contains the following logical logs:

 4

Program over.
$
```

Figure 190. Use INFORMIX-OnLine ontape for Level-0 Backup of the Database

The level-0 backup completed successfully, so we now back up the /home/informix6/tapedev file using the ADSM backup/archive client. Figure 191 shows the main ADSM window for the backup/archive client.

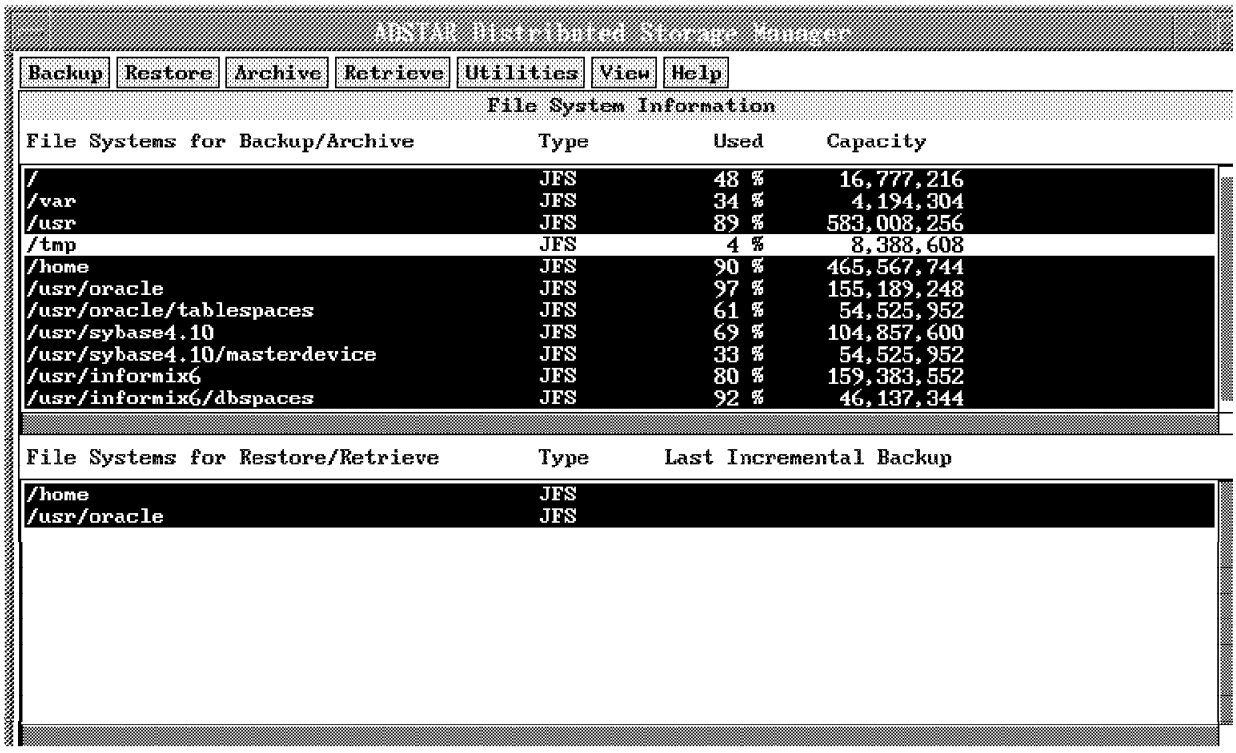


Figure 191. Use ADSM to Back Up the Output of INFORMIX-OnLine ontape

Click on Backup and select the Backup by file specification option (not shown). The ADSM - Backup Specification window appears (see Figure 192 on page 353).

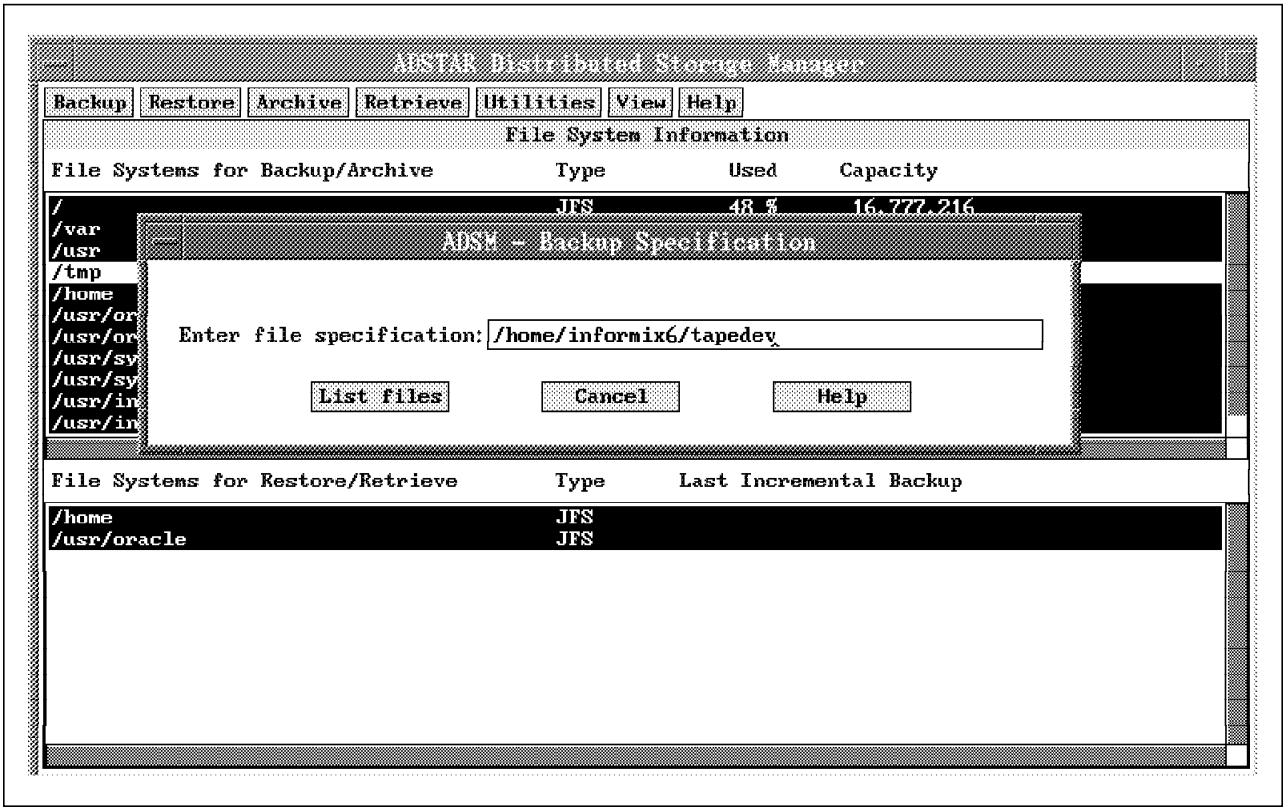


Figure 192. Specify INFORMIX-OnLine File for ADSM Backup

Enter the file name of the file created by the ontape utility, /home/informix6/tapedev, and click on List files. The Backup by File Specification window appears (see Figure 193 on page 354). Click on Select All and then on Backup.

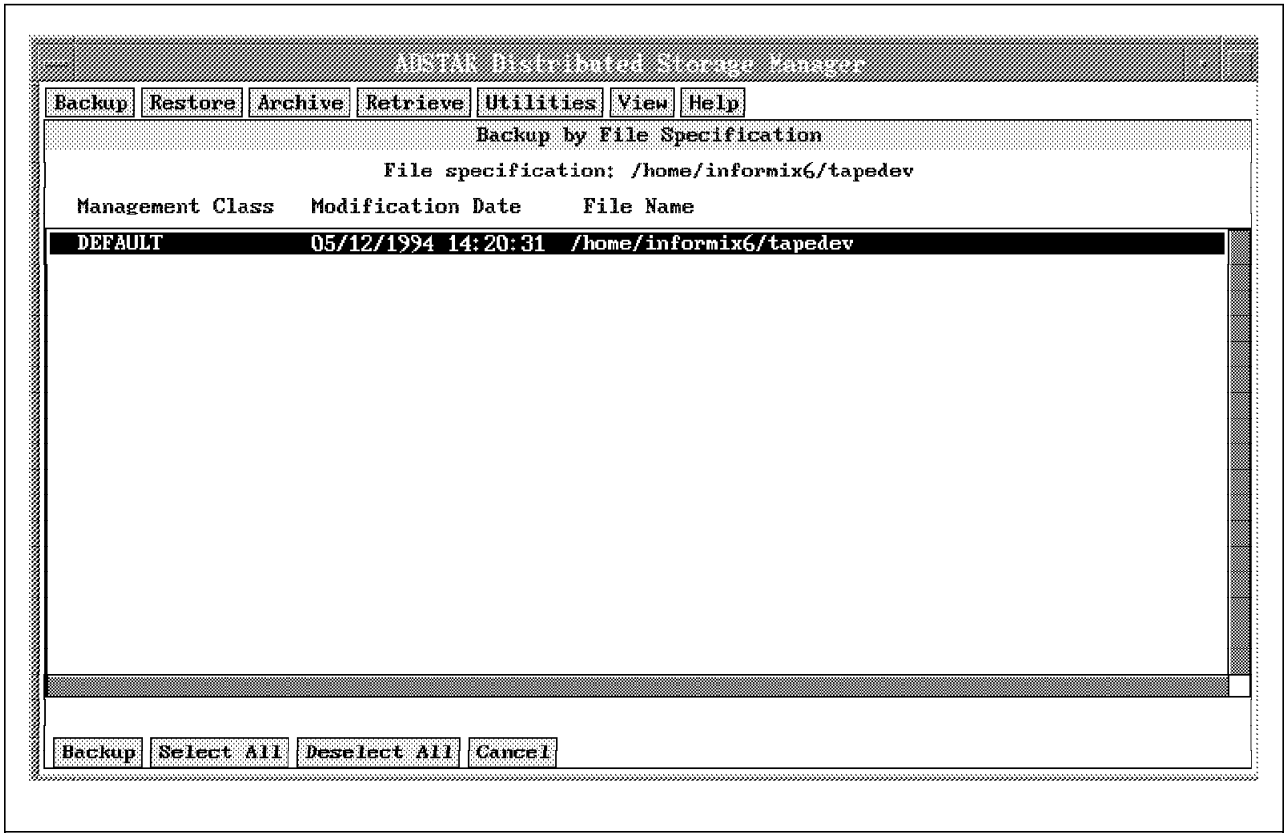


Figure 193. Select INFORMIX-OnLine File for ADSM Backup

The Backup Status window appears (see Figure 194 on page 355). ADSM has now stored your INFORMIX-OnLine Version 6.0 level-0 backup.

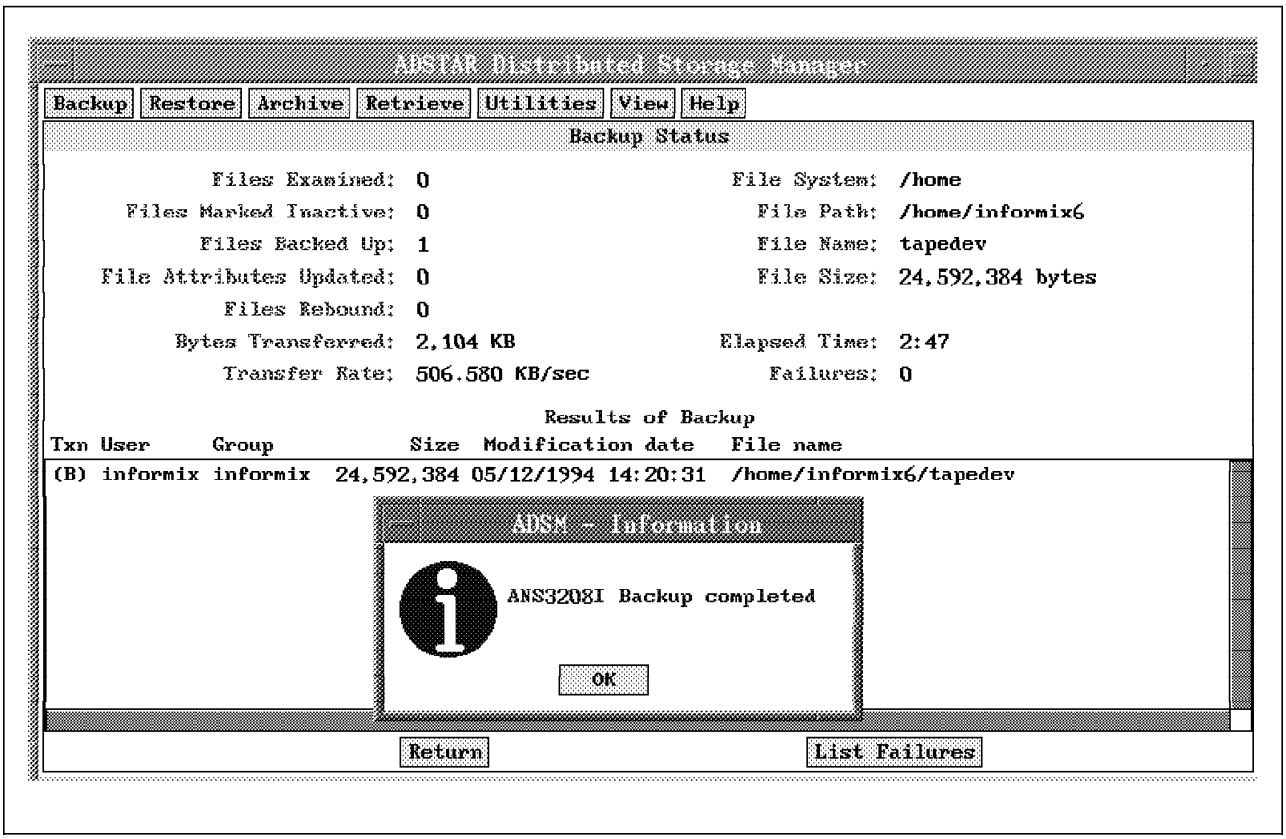


Figure 194. ADSM Backup of INFORMIX-OnLine ontape Output Is Complete

To restore a specific dbspace, first restore the file from ADSM and then use ontape to do a warm restore. To restore the /home/informix6/tapedev file using the ADSM GUI elect to show only the active files (from the ADSM dsm View window - not shown). Now when you click on Restore and select Restore by file specification from the pull-down menu (not shown), you will get the ADSM - Restore Specification window (see Figure 195 on page 356).

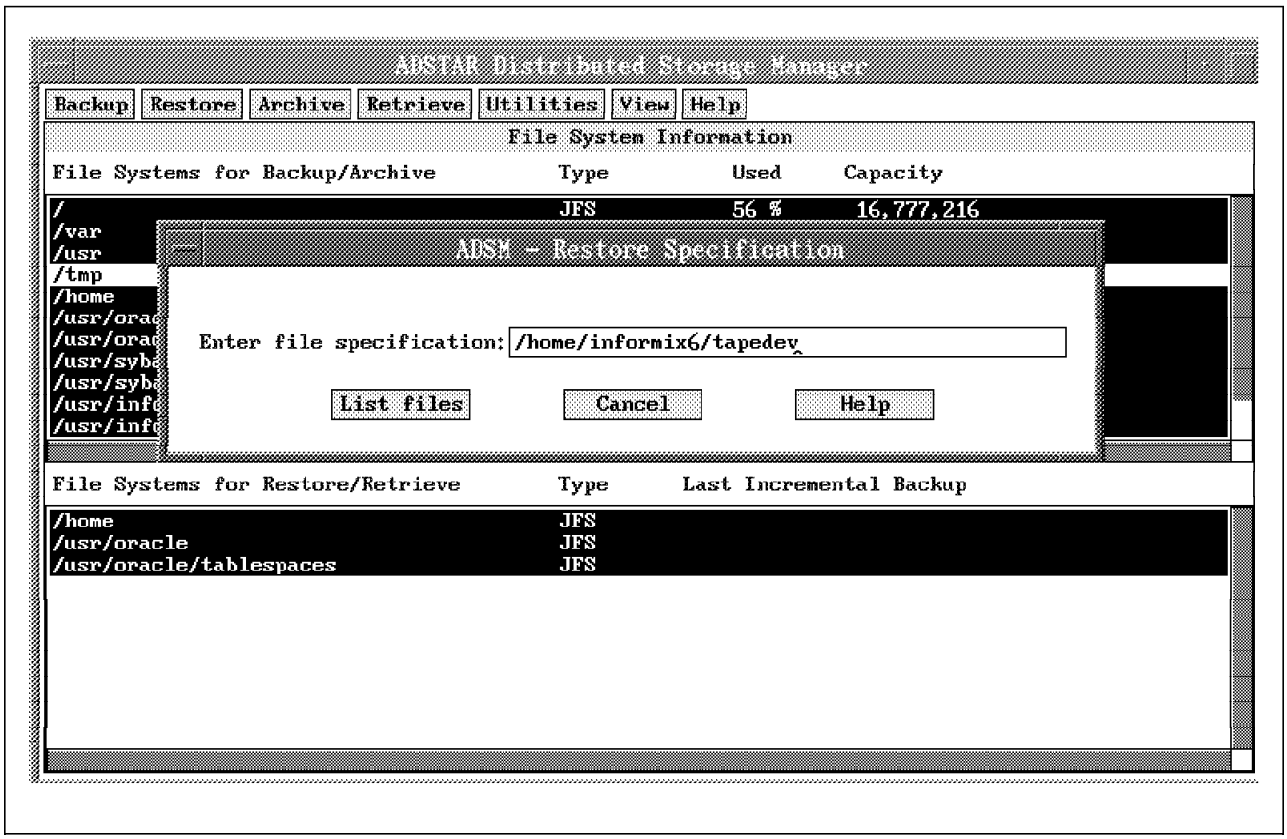


Figure 195. Use ADSM to Restore INFORMIX-OnLine ontape Output

Enter the file name of the ontape file created, /home/informix6/tapedev, and click on List files. The Restore by File Specification window appears (see Figure 196 on page 357).

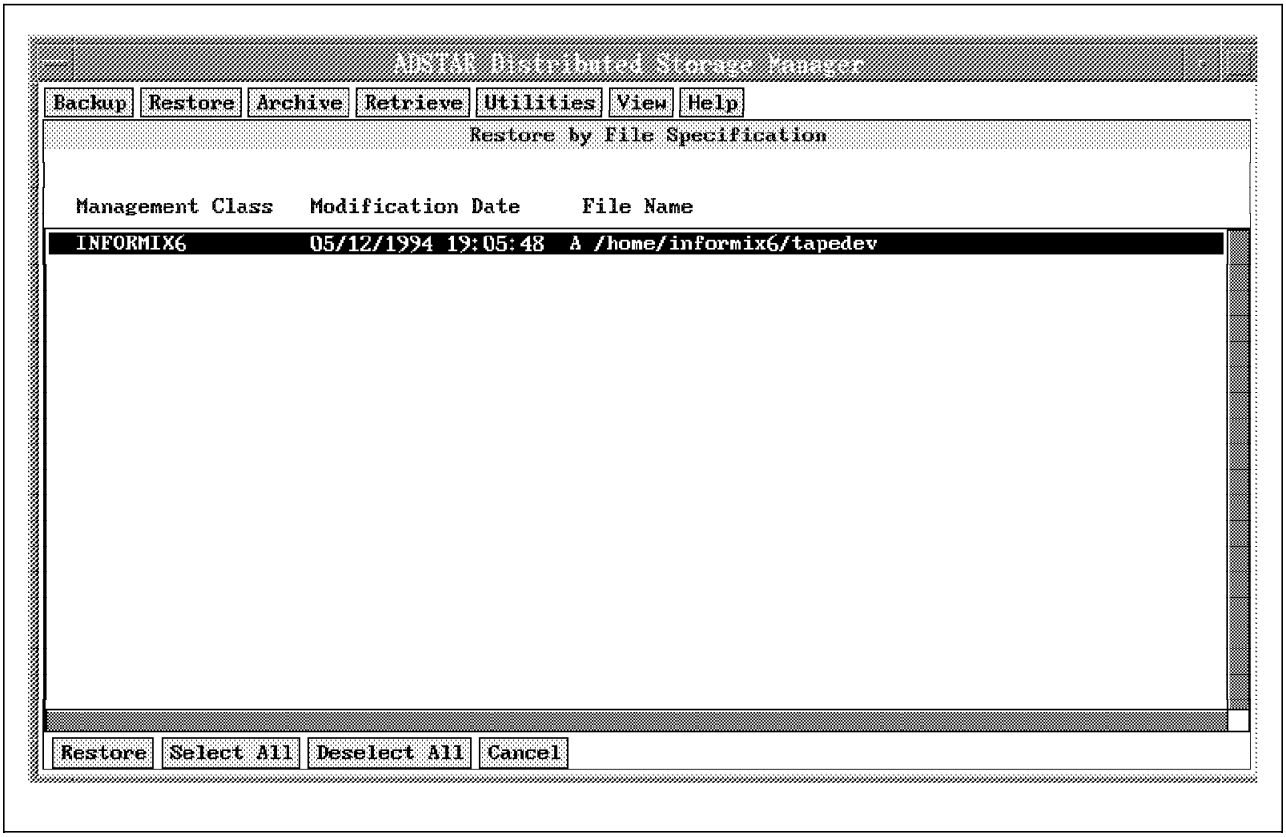


Figure 196. Select the INFORMIX-OnLine ontape Output File for Restore

Click on Select All and Restore to get the ADSM - Restore/Retrieve Parameters window (see Figure 197 on page 358).

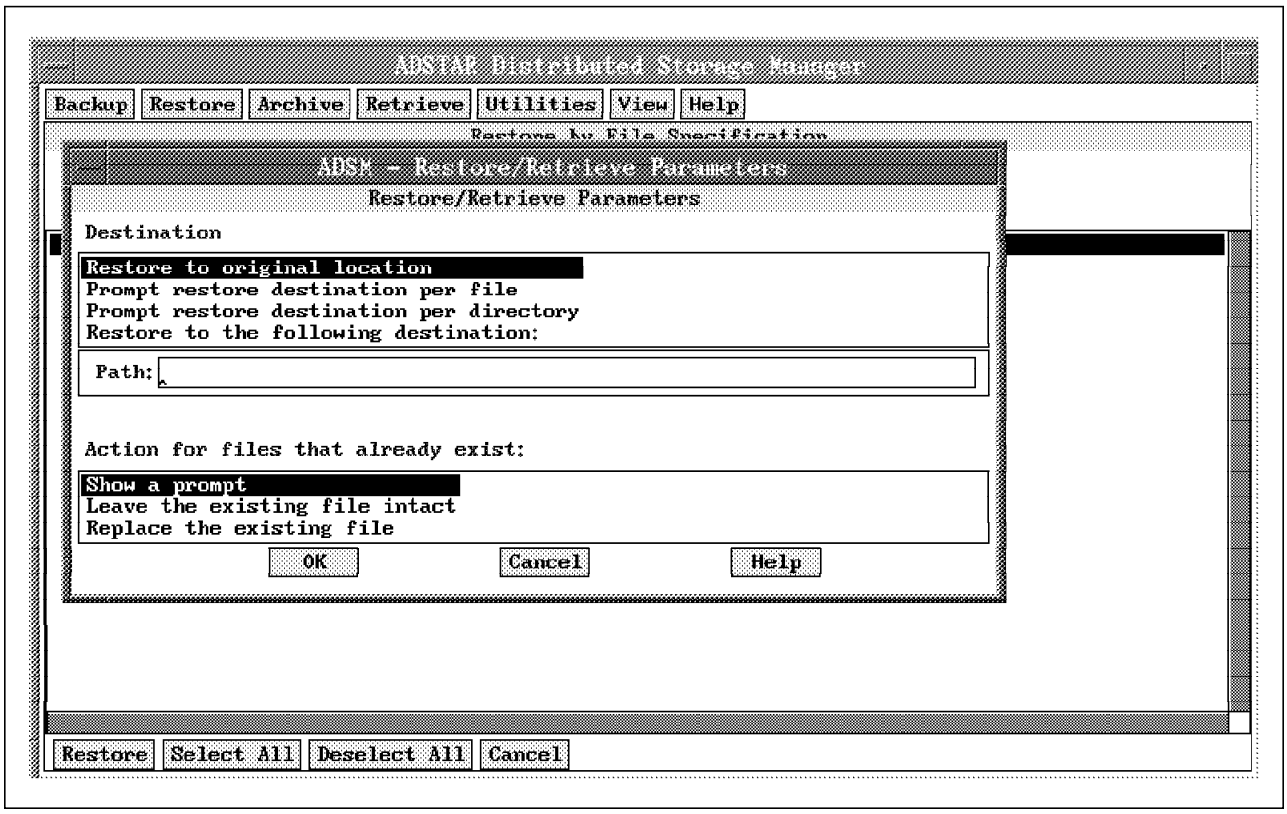


Figure 197. Restore INFORMIX-OnLine ontape Output File to Original Location

Click on OK, and the ADSM-Collision window (Figure 198 on page 359) appears to warn you that a collision has occurred. This means that the original file still exists on your workstation, so you can choose how to proceed.

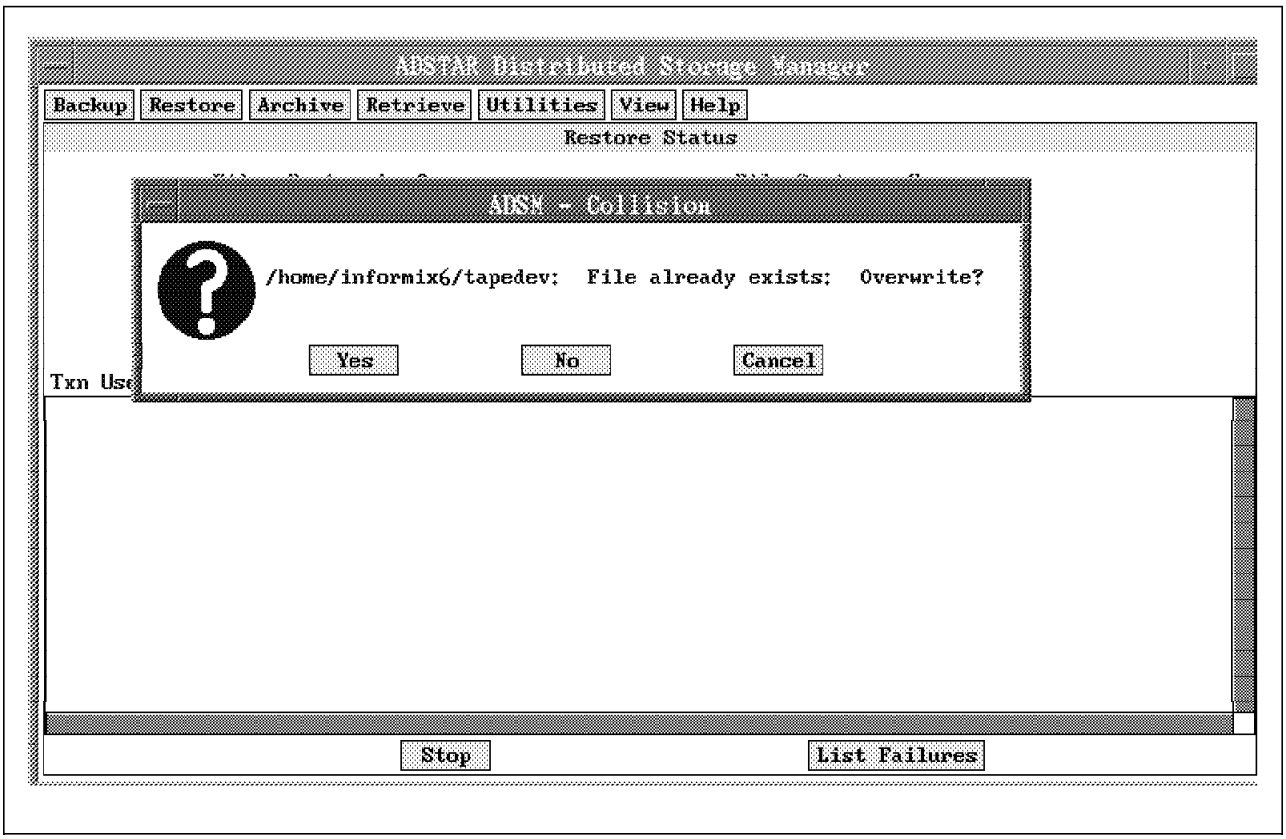


Figure 198. ADSM - Collision Window for INFORMIX-OnLine Restore

We choose to overwrite the original file on the workstation by clicking on Yes. Then the *Restore Status* window appears (Figure 199 on page 360), and the file is restored from ADSM successfully.

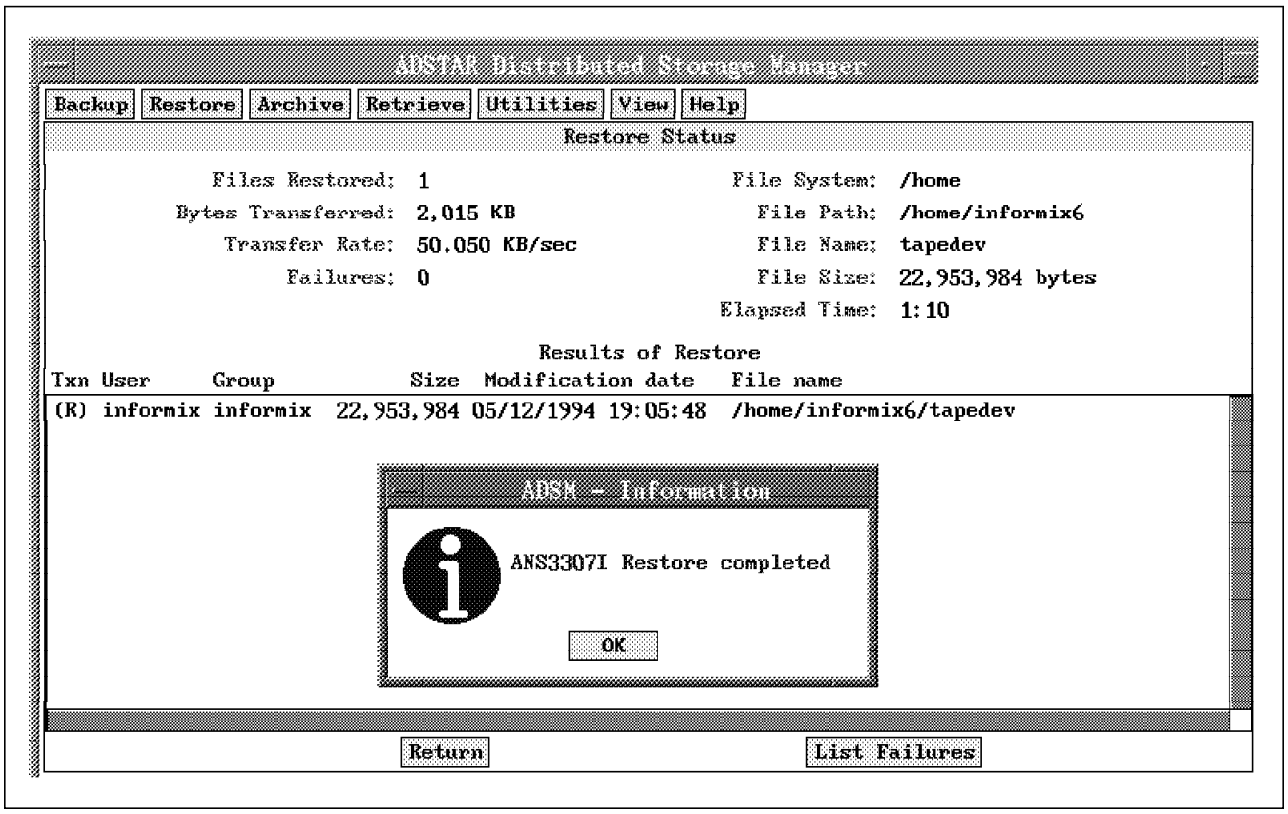


Figure 199. Restore of INFORMIX-OnLine ontape Output Is Complete

Now we are ready to complete the restore process using ontape warm restore. Let us assume that the in6fs dbspace has been destroyed, so we only need to restore this particular dbspace. We can use warm restore because the inf6fs dbspace does not contain RDBMS relevant information, such as logical logs, and it is not the rootdbs dbspace. Figure 200 on page 361 shows the ontape command.

```
$ ontape -r -D inf6fs

DBSpace 'inf6fs' is online; restoring 'inf6fs' will bring all chunks
comprising the DBSpace OFFLINE and will terminate all active
transactions and queries accessing the DBSpace.

OK to continue? yes

Please mount tape 1 on /home/informix6/tapedev and press Return to continue ...

Archive Tape Information
Tape type:      Archive Backup Tape
Online version: INFORMIX-OnLine Version 6.00.UD1
Archive date:   Sat May 14 18:25:27 1994
User id:        informix
Terminal id:    /dev/pts/0
Archive level:  0

Tape device:    /home/informix6/tapedev
Tape blocksize (in k): 16
Tape size (in k): 50000
Tape number in series: 1
Continue restore? (y/n) y

Spaces to restore:1 (inf6fs            )

Restore a level 1 or 2 archive (y/n) n
```

Figure 200. Restore the inf6fs dbspace Using INFORMIX-OnLine ontape

If you want to restore logical log files, you must first restore them from the ADSM server. Because the procedure to restore logical logs is nearly the same as our tbtape example, we do not explain the necessary steps in detail at this point. However, Figure 201 shows the rest of the ontape session, assuming you had restored the logical log files from ADSM first.

```
Do you want to restore log tapes? (y/n) y

Roll forward should start with log number 11

Please mount tape 1 on /home/informix6/ltapedev and press Return to continue ..

Do you want to restore another log tape? (y/n) n

Program over.
$
```

Figure 201. Roll-Forward Recovery of INFORMIX-OnLine Logical Logs

11.3.5 Full ′Almost-Offline′ Backup Using ON-Archive and ADSM

As discussed in 11.2, “INFORMIX-OnLine Backup Utilities” on page 329, ON-Archive is a powerful new utility for INFORMIX-OnLine Version 6.0. It provides online and quiescent mode backup and online and offline restore for one or more dbspaces in parallel. You group one or more dbspaces into dbspace sets, and ON-Archive works against those sets.

If you want to perform backup and restore activities with onarchive, you must define requests and then execute them. In the sections that follow, we describe how to set up dbspace sets and requests with ON-Archive, perform a full database server level-0 and level-1 backup storing the results in ADSM storage, perform a cold (offline) restore of the entire database server, back up the logical logs, and perform a warm (online) restore of a dbspace set.

11.3.5.1 ON-Archive Setup

To set up your ON-Archive environment, you can use the ON-Archive menu interface or the command line interface or issue the commands directly from the AIX command line. Figure 202 shows an ON-Archive command line interface session where the subcommands of ON-Archive are issued within the ON-Archive utility. To set up our environment we defined:

- A volume set with the device type disk
- A volume with the /home/informix6/vols destination directory
- A dbspace set with the rootdbs dbspace
- A dbspace set with the inf6fs and inf6fs2 dbspaces.

```
$ onarchive

onarchive Version 6.0
Copyright (c) 1993 - Informix Software, Inc.

Onarchive> define/vset=inf6adsm/accessibility=10,class=system,driver=disk/device_type=disk

Vset inf6adsm created.
Onarchive> define/volume/vset=inf6adsm/max_space=200000/virtual=(/home/informix6/vols

ARC-W-01844, WARNING: Directory is invalid or does not exist.
Volume number 0001 created in inf6adsm.
Onarchive> define/dbspaceset=rootdbs_set/dbspace=(rootdbs)

    ′Success′

    Define Dbspace Set : rootdbset

Onarchive> define/dbspaceset=inf6fs_set/dbspace=(inf6fs,inf6fs2)

    ′Success′

    Define Dbspace Set : inf6fsset

Onarchive> exit
```

Figure 202. INFORMIX-OnLine Onarchive Setup Commands

11.3.5.2 Full Database Server Level-0 Backup Stored in ADSM

To perform a level-0 backup, you must define a request. Figure 203 shows how to:

- Define a level-0 archive request that includes all dbspace sets
- Execute the request
- List the status of the executed request.

```
$ onarchive 'archive/dbspaceset=*/level=0/vset=inf6adsm'

Request 00000001 registered in the catalog.

$ onarchive 'execute/request=1'
Executing...
File rootdbs being processed.
File inf6fs             being processed.
File inf6fs2            being processed.

$ onarchive 'list/request=*'
RID       Username           Command    Issue Date   Status       Former RID/SID

--------  -----------------  ---------  -----------  -----------  ---------- -- -
00000001  informix           ARCHIVE    19-May-1994  COMPLETED    ********   ***

Total of 1 request(s)
```

Figure 203. Level-0 Backup Using INFORMIX-OnLine ON-Archive

You can list the files that ON-Archive has created in the /home/informix6/vols directory as shown in Figure 204.

```
$ ls -al /home/informix6/vols
total 48040
drwxrwxrwx   2 informix informix     512 May 19 16:14 .
drwxr-xr-x   3 informix informix     512 May 19 15:16 ..
-rw-rw----   1 informix informix 24547182 May 19 16:15 00000001.SAV
-rw-r--r--   1 informix informix      88 May 19 15:16 VOL.HDR
$
```

Figure 204. Output from INFORMIX-OnLine ON-Archive Command

The 00000001.SAV file contains the level-0 archive. The file name includes the INFORMIX-OnLine internal request ID, which is also the saveset ID you would specify when you restore the level-0 archive. The ON-Archive utility also updated the volume header file, VOL.HDR, which contains the name of the vset (inf6adsm, in our example) and the number of volumes.

To back up the contents of the /home/informix6/vols directory, including the level-0 archive and volume header file, you can use the ADSM dsmc selective command. Figure 205 on page 364 shows the ADSM dsmc selective command to back up the files of the /home/informix6/vols directory.

```
$ dsmc selective -password=mars -quiet /home/informix6/vols/*
ADSTAR Distributed Storage Manager
Command Line Backup Client Interface - Version 1, Release 2, Level 0.0
(C) Copyright IBM Corporation, 1990, 1994, All Rights Reserved.
```

Figure 205. ADSM Command to Back Up INFORMIX-OnLine ON-Archive Output

11.3.5.3 Full Database Server Level-1 Backup Stored in ADSM

If you want to perform a level-1 archive on the next day, you define another request with ON-Archive and execute the request. Figure 206 shows these steps:

```
$ onarchive 'archive/dbspaceset=*/level=1/vset=inf6adsm'

Request 00000003 registered in the catalog.

$ onarchive 'execute/request=3'
Executing...
File rootdbs being processed.
File inf6fs             being processed.
File inf6fs2            being processed.
$
```

Figure 206. INFORMIX-OnLine ON-Archive Command for Level-1 Backup

The ON-Archive utility created a file called /home/informix6/vols/00000003.SAV and updated the volume header file, VOL.HDR. Figure 207 shows the ADSM dsmc selective commands to back up both the /home/informix6/vols/00000003.SAV and /home/informix6/vols/VOL.HDR files.

```
$ dsmc selective -password=mars -quiet /home/informix6/vols/00000003.SAV

ADSTAR Distributed Storage Manager
Command Line Backup Client Interface - Version 1, Release 2, Level 0.0
(C) Copyright IBM Corporation, 1990, 1994, All Rights Reserved.

$ dsmc selective -password=mars -quiet /home/informix6/vols/VOL.HDR

ADSTAR Distributed Storage Manager
Command Line Backup Client Interface - Version 1, Release 2, Level 0.0
(C) Copyright IBM Corporation, 1990, 1994, All Rights Reserved.
```

Figure 207. ADSM Backup of INFORMIX-OnLine ON-Archive Level-1 Output

11.3.5.4 Cold Restore of the Entire Database Server

Here is how to perform the corresponding restore activity with the ADSM backup/archive client GUI and the ON-Archive utility. Figure 208 on page 365 shows the File System Information window, with the /home/informix6/vols directory highlighted.

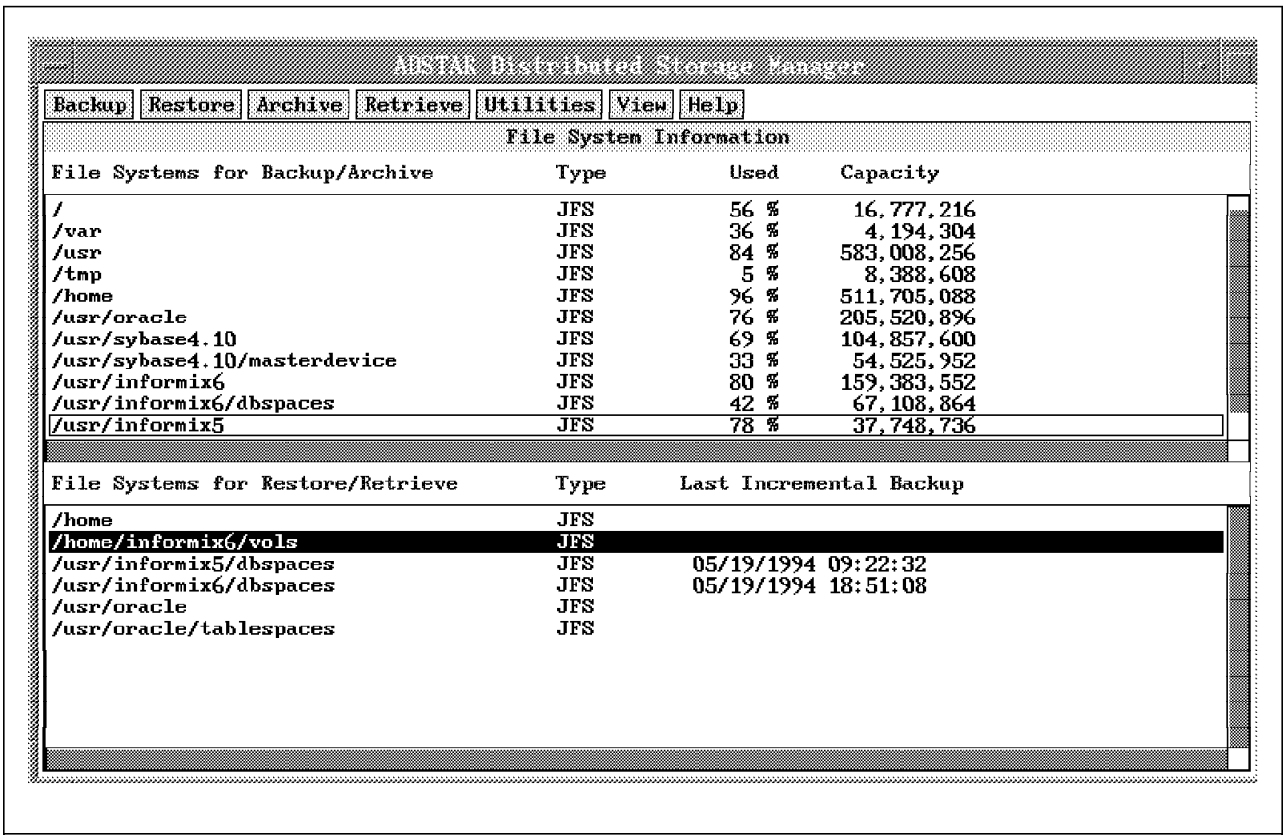


Figure 208. ADSM Restore of INFORMIX-OnLine Database Server

Click on Restore and select Restore by file specification from the pull-down menu (not shown). The ADSM - Restore Specification window appears (Figure 209 on page 366).

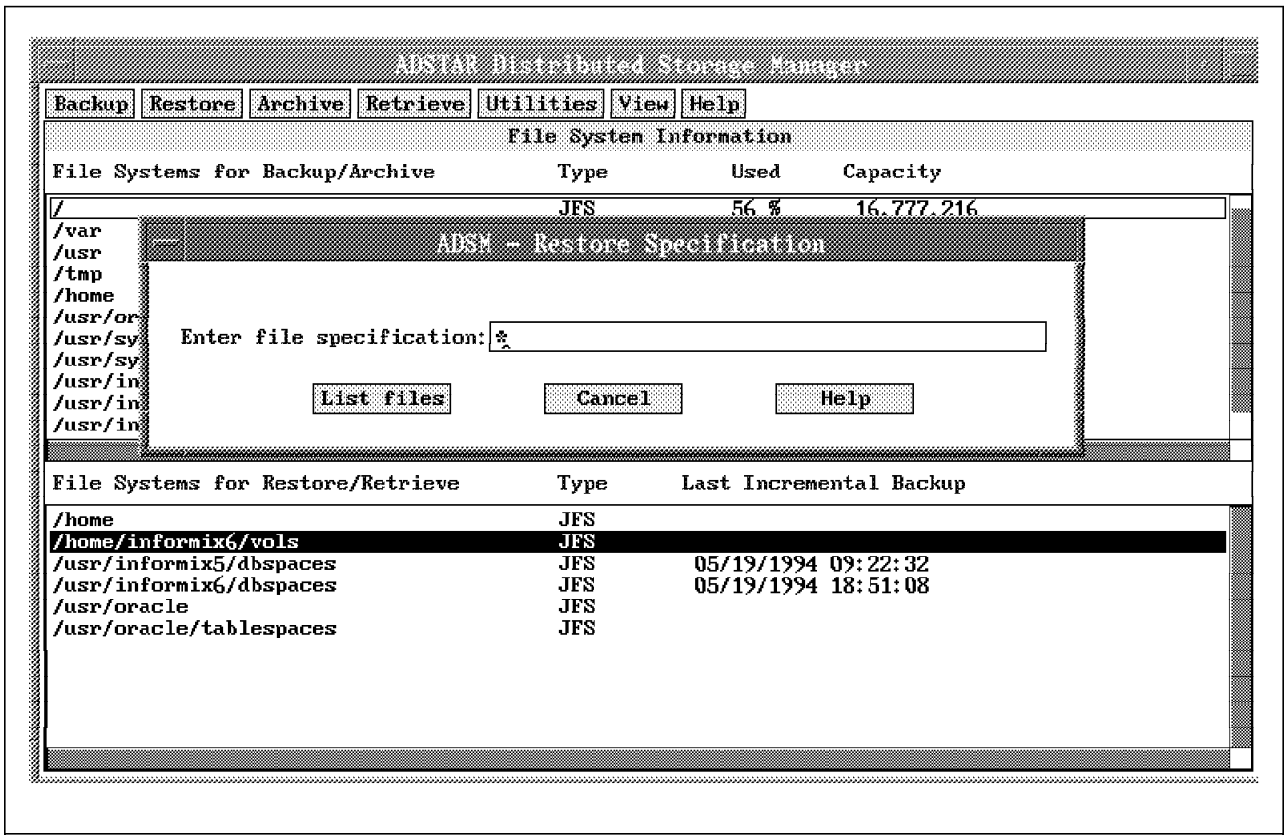


Figure 209. List All INFORMIX-OnLine Files That Can Be Restored

Enter an asterisk in the Enter file specification field to list all files and click on List files. The Restore by File Specification window appears with a list of three files (Figure 210 on page 367).

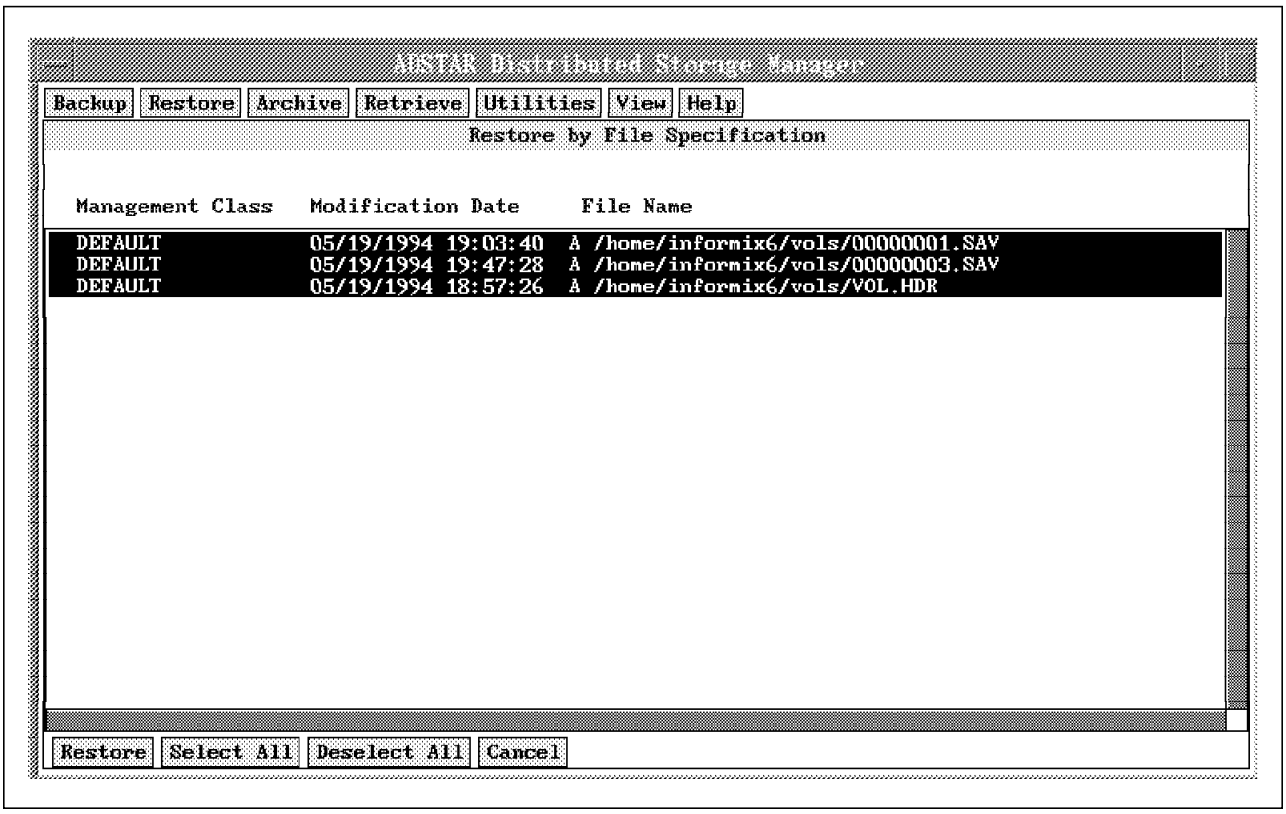


Figure 210. Select All INFORMIX-OnLine Files for Restore

Click on Select All to highlight all three files, and then click on Restore to initiate the restore process. The ADSM - Restore/Retrieve Parameters window appears (Figure 211 on page 368).

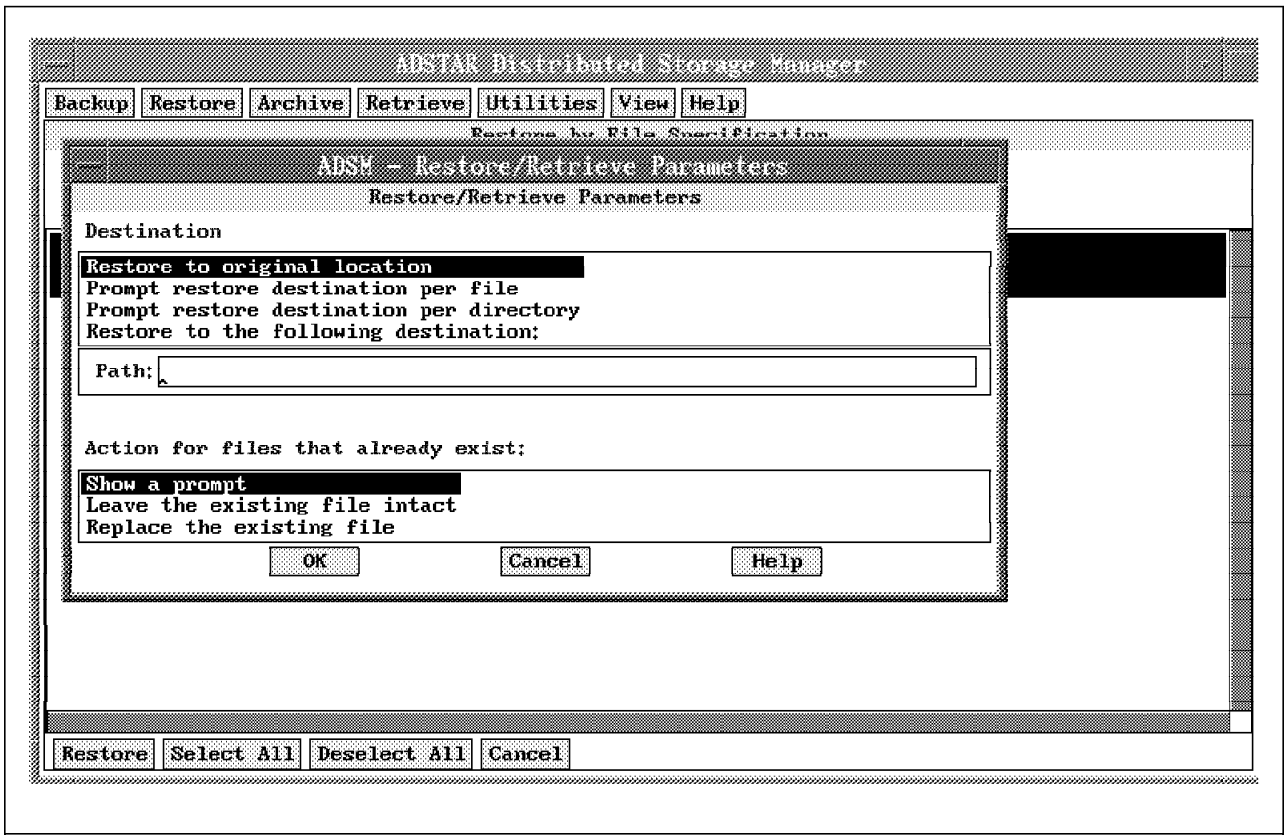


Figure 211. Restore INFORMIX-OnLine ON-Archive Output to Original Location

Click on OK, and the Restore Status window appears (Figure 212 on page 369). The ADSM part of the restore process is now complete.

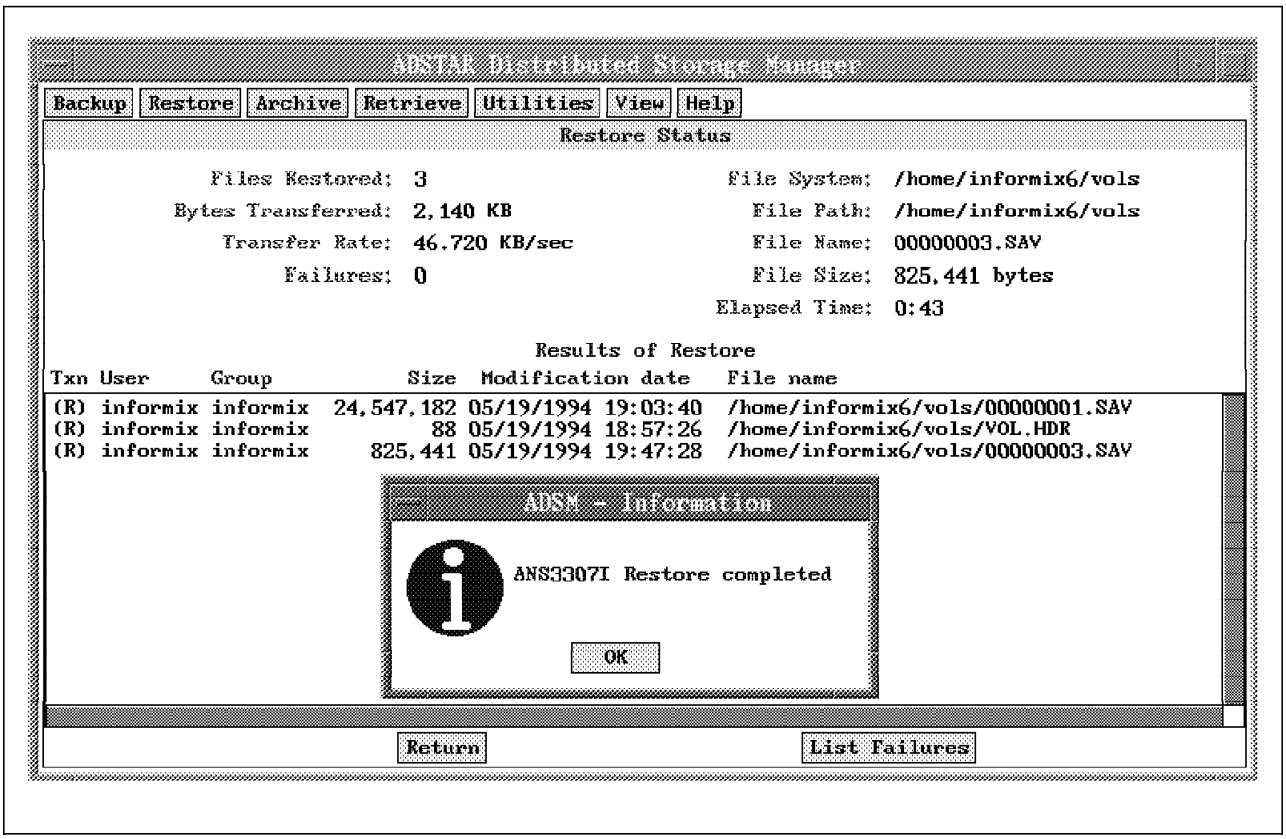


Figure 212. ADSM Part of INFORMIX-OnLine Restore Process Is Complete

Next, we finish restoring our database server using INFORMIX-OnLine Version 6.0 utilities. The utilities you use depend on which dbspace needs to be restored and whether you perform a warm or cold restore:

- When the rootdbs dbspace or logical logs do not need to be restored, you can perform a warm restore by using the ON-Archive utility.
- When the rootdbs dbspace or logical logs must be restored, you must perform a cold restore by using either the ondatartr utility or the ondatartr utility in combination with the ON-Archive utility.

In our example, we assume that the rootdbs dbspace has been destroyed, and we need to perform a cold restore using the ondatartr utility. Figure 213 on page 370 shows the ondatartr utility session to restore an entire database server with a cold restore.

```
$ ondatartr 'retrieve/dbspaceset=*/disk=(/home/informix6/vols)/salvagelogs=(/home/informix6/vols)'

                onarchive: Media Identification Function

Disk Information:

                Vset       : inf6adsm
                Volume     : 000001
                Virtual    : /home/informix6/vols

ARC-W-01845, Initializing this media may overwrite an ON-Archive volume.

Proceed and overwrite contents, or Cancel operation?
Enter (P/C): p
Log files 12 through 12 are eligible for salvaging.
What log file number should ONDATARTR start with? 12
Logfile LF00000012 being processed.
Executing...
```

Figure 213. INFORMIX-OnLine Ondatartr Process for Cold Restore

The ondatartr utility saves the current logical log. In Figure 213, the LF00000012 log file was saved. You are then prompted for the request ID (which is also the saveset ID). In our example, our saveset ID for the level-0 archive is 1: /home/informix6/vols/00000001.SAV. Figure 214 shows how the ondatartr restore process continues.

```
What saveset ID is to be used on volume /home/informix6/vols? 1

Level 0 physical restore started.

The file rootdbs has been retrieved.

The file inf6fs             has been retrieved.

The file inf6fs2            has been retrieved.

End of saveset reached on this volume.
Does this saveset continue on another volume? (Y/N) n
```

Figure 214. Enter saveset ID to Continue INFORMIX-OnLine ondatartr Process

Please note that in our example the logical restore began with the salvaged log, log file 12. However, the logical restore really begins with the log that holds the checkpoint from the last-level archive (level-1, in our example). In most cases, this means that multiple log files will need to be entered as input to ondatartr, not just one log file.

For example, let us assume that the user did a level-0 and level-1 archive and that during the level-1 archive, log file 8 was current. When the user issues the retrieve logfile command, they are told that the logical restore is started with log number 8. The user must then tell ondatartr the disk path and saveset ID for EACH log file between 8 and 12. This requires manual input, but in the future a more automated approach will be provided.

The next step is to restore the level-1 archive from the /home/informix6/vols directory. The level-1 archive was defined with request number 3, so the corresponding saveset ID is also 3. Figure 215 on page 371 shows the completion of retrieving the archived files.

```
Do you have a level 1 archive to retrieve? (Y/N):y
Enter the directory to be used next: /home/informix6/vols
What saveset ID is to be used on volume /home/informix6/vols?:3
Level 1 physical restore started.

The file rootdbs has been retrieved.

The file inf6fs             has been retrieved.

The file inf6fs2            has been retrieved.

End of saveset reached on this volume.
Does this saveset continue on another volume? (Y/N) n
Do you have a level 2 archive to retrieve? (Y/N) n
```

Figure 215. Continue INFORMIX-OnLine ondatartr Process for Level-1 Archive

The ondatartr utility does not prompt you for the previously saved logical log (log file 12). Therefore, to restore logical log file 12, you must enter an additional ondatartr command. To determine which saveset ID the previous ondatartr utility assigned to logical log file 12, you must look at the /home/informix6/vols directory. This directory contains an additional file that the ondatartr utility created, with a saveset ID of 00017759 and a file name of 00017759.SAV. This file needs to be restored. Figure 216 shows the restore of the logical logs.

```
$ ondatartr 'retrieve/logfile/disk=(/home/informix6/vols)'
Executing...
Logical restore started with log number: 12.
What saveset ID is to be used on volume /home/informix6/vols? 17759

The file LF00000012 has been retrieved.

End of saveset reached on this volume.
Does this saveset continue on another volume? (Y/N) n
Do you have more log backups to process? (Y/N) n
```

Figure 216. Use INFORMIX-OnLine ondatartr to Restore Logical Logs

11.3.5.5 Continuous Backup of Logical Logs with ON-Archive and ADSM

Here is an example of a continuous backup of the logical logs with ON-Archive and ADSM. Remember that continuous logging means that the logs are automatically backed up when they are filled. The procedure for defining continuous backups for the logical log is the same as the procedure for defining other requests. Also, you need a dedicated terminal or window because this process cannot run in the background. Figure 217 on page 372 shows how to define and execute the request.

```
$ onarchive 'backup/logfile/continuous/current/vset=inf6adsm'

Request 00000005 registered in the catalog.

$ onarchive 'execute/request=5'
Executing...
```

Figure 217. Use INFORMIX-OnLine ON-Archive to Back Up Logical Logs Continuously

Just as with the other examples, the ON-Archive utility creates a file with a unique saveset ID that corresponds to the request ID.

Because ON-Archive creates each file with a unique saveset ID, you can use the ADSM dsmc incremental command to back up the newly created files in the /home/informix6/vols directory. Figure 218 shows the appropriate ADSM dsmc incremental command to back up the newly created file.

```
$ dsmc incremental -password=mars -quiet /home/informix6/vols

ADSTAR Distributed Storage Manager
Command Line Backup Client Interface - Version 1, Release 2, Level 0.0
(C) Copyright IBM Corporation, 1990, 1994, All Rights Reserved.

Session established with server BALTIC: AIX-RS/6000
 Server Version 1, Release 2, Level 0.0
 Server date/time: 05/20/1994 12:41:48   Last access: 05/20/1994 12:40:25

Incremental backup of file system: '/home/informix6/vols'
Normal File-->       2,707,456 /home/informix6/vols/00000005.SAV ......... Sent
```

Figure 218. ADSM Backup of INFORMIX-OnLine Log File

11.3.5.6 Warm Restore of a dbspace Set

If you need to restore a set of dbspaces other than the rootdbs dbspace or the dbspaces that contain the logical logs, you can perform a warm restore with onarchive. You only need to define the requests within ON-Archive to restore a specific dbspace set with a specific dbspace. Figure 219 shows the appropriate command to define the requests to retrieve two dbspaces, inf6fs and inf6fs2, of dbspace set, inf6fs_set.

```
$ onarchive 'retrieve/dbspaceset=inf6fs_set/dbspace=inf6fs'

Request 00000007 registered in the catalog.

$ onarchive 'retrieve/dbspaceset=inf6fs_set/dbspace=inf6fs2'

Request 00000008 registered in the catalog.
```

Figure 219. Define the INFORMIX-OnLine dbspace Set to Retrieve Two dbspaces

You do not need to specify a directory or a file because this information is obtained from the sysmaster tables of the INFORMIX-OnLine database server.

The database server knows which files in the /home/informix6/vols directory contain data so that the dbspaces can be rebuilt. Figure 220 on page 373 shows the execution of the requests to retrieve the inf6fs and inf6fs2 dbspaces.

```
$ onarchive ′execute/request=7′
Executing...

Any online dbspaces being retrieved will first be taken offline.

Do you want to Proceed or Cancel this request? (P/C) p
File inf6fs being retrieved.
File inf6fs being retrieved.

$ onarchive ′execute/request=8′
Executing...

Any online dbspaces being retrieved will first be taken offline.

Do you want to Proceed or Cancel this request? (P/C) p
File inf6fs2 being retrieved.
File inf6fs2 being retrieved.
```

Figure 220. Execute the INFORMIX-OnLine ON-Archive Request to Retrieve Two dbspaces

Please note that to complete the warm restore operation and make the dbspaces available for use you must also roll forward the logical logs. To do this you first back up the current log with the backup logfile command and then roll forward the logical logs with the retrieve logfile command. We do not show an example of this.

11.3.6 Full ′Online′ Backup Using ON-Bar and ADSM

As mentioned in section 11.2.4, “ON-Bar” on page 332, the ON-Bar utility provides significant enhancements over other types of backup and recovery utilities available for use with the INFORMIX-OnLine database solution. The ON-Bar utility allows for online backup of the entire database, dbspaces, blobspaces, and logical log files. ON-Bar, just like ON-Archive, is even flexible enough to allow for “warm” recovery of dbspaces and blobspaces. Our tests included a full online backup and cold recovery of a database, full online backup and online recovery of a dbspace, and continuous, online backup of logical logs. Before we examine individual backup and recovery scenarios, we discuss configuration tasks you must do to ensure interoperability between the ON-Bar product and the ADSM X/OPEN API.

Figure 221 on page 374 documents our testing environment.

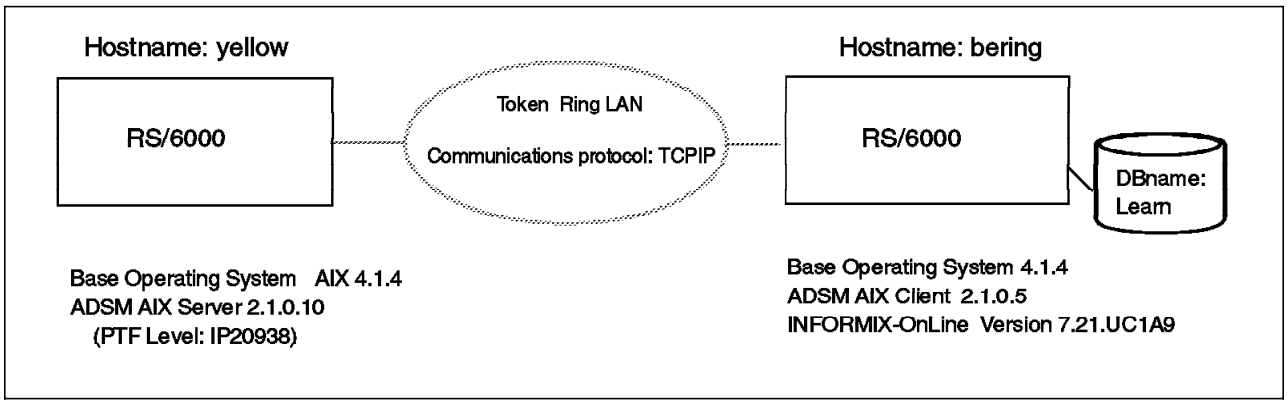


Figure 221. ON-Bar Testing Environment

We used two AIX machines, yellow and bering. Yellow was our ADSM AIX server, and Bering was our INFORMIX-OnLine server. We created one sample database, Learn.

11.3.6.1 ON-Bar and ADSM X/OPEN API Setup

Installation of the ON-Bar utility occurs when INFORMIX-OnLine Version 7.21 is installed. The ON-Bar executable, onbar, is stored in the $INFORMIXDIR/bin directory.

Technical details on the installation of the ADSM client software are described in the *ADSM Installing the Clients* manual. The following components and minimum levels of ADSM client code must be installed to enable ON-Bar function with the ADSM storage manager:

- ADSM Backup/Archive Client - 2.1.0.5
- ADSM Application Programming Interface - 2.1.0.5

The ADSM backup/archive client code provides the basic connectivity files, such as the dsm.opt and dsm.sys files, that are needed to establish a connection between the ON-Bar client and the ADSM server. We explain the changes that must be made to the dsm.sys file in step 2 of our instructions in “Integrating ON-Bar with the ADSM X/OPEN API.”

The ADSM API code provides the API library, libXApi.a, in client code version 2.1.0.5, which links to the ON-Bar X/OPEN library name to enable data passing between ON-Bar and the ADSM storage manager. The ADSM API library is located in the /usr/lpp/adsm/api/bin directory.

Integrating ON-Bar with the ADSM X/OPEN API: Instructions on how to integrate ON-Bar with the ADSM X/OPEN API are documented in the README.API file located in the /usr/lpp/adsm/api/bin directory. Some of the instructions provided are specific to the ADSM client code files, some are specific to ON-Bar files, and others link the two products.

Here are the steps we used to ensure interoperability between the ON-Bar product and the ADSM storage manager. The steps closely parallel, but are an expansion of, the tasks outlined in the README.API file included with ADSM client code at the 2.1.0.5 level. Instructions in the README.API file may differ at other levels of ADSM client code. All of the steps are executed by the root user.

1. Define the symbolic link between the ADSM X/Open library name and the ON-Bar X/Open library name to enable data passing.

a. Change the directory to the location where the ADSM API code and README.API files are stored.

```
cd /usr/lpp/adsm/api/bin
```

b. Link the existing ADSM API X/OPEN library to the ON-Bar X/OPEN library:

```
ln -s /usr/lpp/adsm/api/bin/libXApi.a /usr/lib/ibsad001.a
```

Note: The ibsad001.a file, the ON-Bar X/Open library name, DOES NOT exist in the /usr/lib directory until you execute the above link command.

2. Edit the dsm.sys file to set passwordaccess to generate.

Note: The default setting in the dsm.sys file for passwordaccess is implicitly set to prompt. ON-Bar requires that passwordaccess be explicitly set to generate because the ON-Bar utility cannot get or store password data.

3. Configure an ON-Bar file to reflect the level of ADSM client code installed.

Copy the $INFORMIX/etc/sm_versions.std file to sm_versions and edit it by inserting *1|2.1.5|adsm|2*. The | character is the pipe character. The sm_versions.std file, by default, contains only one line. You can either leave the one line in the file and insert *1|2.1.5|adsm|2* as a second line, or you can overwrite the one line with *1|2.1.5|adsm|2*.

Note: The second field, *2.1.5*, references the level of ADSM client code. In our scenario, the ADSM client code was at the 2.1.0.5 level, and the ADSM server code was at the 2.1.0.10 level. The ON-Bar utility references or "knows" the ADSM client code version according to the first, second, and last fields of the IBM fix level output, shown with the *lslpp -h* command.

4. Set the following environment variable:

```
INFORMIXSERVER=servername;export INFORMIXSERVER
```

Note: The variable must be set so that the bldutil.sh script references the correct database server name. The variable is included in the informix userid's .profile but, because the root user runs the bldutil.sh script, the variable is not part of the standard environment and must therefore be added before the script can execute properly.

5. Set the API environment variables:

```
DSMI_DIR=/usr/lpp/adsm/bin
DSMI_LOG=/usr/lpp/adsm/bin
DSMI_CONFIG=/usr/lpp/adsm/bin/dsm.opt
```

Note: These variables are specific to the ADSM X/OPEN API. The informix userid's profile should contain these environment variables. The variables must be set from the command line for the script to run correctly because the root user will run the bldutil.sh shell script in step 6.

6. If you are adding the ON-Bar ADSM support to an existing system you must also SQL insert the following into the bar_version table. Do this through dbaccess.

```
insert into bar_version values ('1','2.1.5','adsm','2');
```

If your database engine has not been started for the first time, that is, you have not initialized it with an *onit -i* command, you do not need to do this SQL insert.

To verify connectivity between ON-Bar and the ADSM storage manager, start a session on the ON-Bar machine as the informix user and issue the **onbar --** command. The command returns usage information specific to onbar. If the command does not return usage information, look at the bar_act.log to see whether a connectivity problem exists between ON-Bar and the ADSM storage manager. Figure 222 shows the output of the **onbar --** command.

```
 1996-10-30 16:17:47 38342  32704 onbar --
 1996-10-30 16:17:47 38342  32704 onbar usage

BACKUP:
=======
-b [-L <level>] [-f <filename>] [<dbspace list>]
-b -w [-L <level>]
-b -F
-l [-c] [-s]
    -b backup
    -c backup current logical log
    -f pathname of file containing list of dbspaces, blobspaces
    -F fake backup
    -l backup full logical logs
    -L backup level: 0, 1, or 2
    -w whole system backup
    -s salvage logs
    <dbspace list> list of dbspaces, blobspaces to backup

RESTORE
========
-r [-t <time> | -n <log>] [-f <filename>] [<dbspace list>]
-r -p [-t <time>] [-f <filename>] [<dbspace list>]
-r -l [-t <time> | -n <log>] [-f <filename>] [<dbspac e list  >]
-r -w [-p] [-t <time> | -n <log>]
    -f pathname of file containing list of dbspaces, blobspaces
    -l logical restore
    -n last logical log to restore
    -p physical restore
    -r restore
    -t point in time to stop restore.
    <dbspace list> list of dbspaces, blobspaces to restore
    -w whole system to restore
```

Figure 222. Output of the onbar -- Command

If the **onbar --** command does not work, check that your environment variables are set properly. Figure 223 on page 377 shows the environment variable output from our informix userid.

```
_=/usr/bin/env
LANG=en_US
LOGIN=informix
NLSPATH=/usr/lib/nls/mbs/%L/%N:/usr/lib/nls/mgs/%L/%N.cat
PATH=/usr/bin:/etc:/usr/sbin:/usr/ucb:/home/informix/bin:
     /usr/bin/X11:/sbin:.:/usr/informix/7.2/bin
LC_FASTMSG=true
INFORMIXSQLHOSTS=/usr/informix/7.2/etc/sqlhosts
LOGNAME=informix
MAIL=/usr/spool/mail/informix
DSMI_CONFIG=/usr/lpp/adsm/bin/dsm.opt
LOCPATH=/usr/lib/nls/loc
USER=informix
AUTHSTATE=compat
SHELL=/usr/bin/ksh
ODMDIR=/etc/objrepos
DSMI_LOG=/usr/lpp/adsm/bin
HOME=/home/informix
INFORMIXDIR=/usr/informix/7.2
TERM=lft
MAILMSG=
ONCONFIG=onconfig.learn
PWD=/home/informix
TZ=PST8PDT
DSMI_DIR=/usr/lpp/adsm/bin
INFORMIXSERVER=learn
A_z=! LOGNAME
```

Figure 223. Informix User ID Environment Variables

Note

We describe two ON-Bar options in more detail: -w and -F

The -w, whole system backup, option backs up all dbspaces as a self-contained, restorable unit. It is not the same as backing up all dbspaces. The -w option also backs up extra information which it imbeds in the actual dbspaces. A whole system restore can only be done from a whole system backup. A logical restore is not required, but is the command default if you do not specify additional options. The -w option is functionally synonymous with OnTape. The drawback of using the -w option is that parallel backups and restores are not permitted.

The -F, fake backup, option can be used to facilitate changes in database logging modes; to allow the user to use new logs, chunks, or mirrors without performing a backup; or in special situations when the administrator judges that a backup is not needed. A backup does not actually occur, so a restore from a fake backup is not possible. The -F option puts a list of all dbspaces (that you select to back up) in a file, instead of listing them on the command line. INFORMIX recommends that you use fake backups sparingly, if at all.

For more information, see the *INFORMIX-OnLine Archive and Backup Guide*.

ON-Bar Tuning and Configuration Parameters: ON-Bar stores configuration information in the INFORMIX-OnLine *$ONCONFIG* file. If a parameter is not set in the $ONCONFIG file, ON-Bar uses a default value.

The ON-Bar parameters are:

- BAR_ACT_LOG

  The BAR_ACT_LOG parameter is the full path name of the activity message file that ON-Bar uses to record information about its activities. The default is /tmp/bar_act.log.

- BAR_MAX_BACKUP

  The BAR_MAX_BACKUP parameter is the maximum number of parallel sessions that ON-Bar can use to back up data to the ADSM storage manager per ON-Bar call. For example, if BAR_MAX_BACKUP is 5 and ON-Bar is called twice, the maximum number of sessions to the ADSM storage manager is 10. If the BAR_MAX_BACKUP parameter is not set or is 0, the same level of parallelism that the ADSM storage manager uses will be used by ON-Bar. The default value is 0.

- BAR_RETRY

  The BAR_RETRY parameter specifies how ON-Bar is to react if an error occurs. If BAR_RETRY is set to BAR_ABORT, ON-Bar aborts the entire backup or restore and returns an error. If BAR_RETRY is set to BAR_CONT, ON-Bar returns an error for the object that caused the error but continues to back up or restore any remaining objects. If BAR_RETRY is set to a numeric value *n*, ON-Bar attempts to back up an object *n* times before moving on to remaining objects. The default is BAR_CONT.

  If a logical log backup or restore fails, ON-Bar aborts the operation regardless of the setting of BAR_RETRY.

- BAR_NB_XPORT_COUNT

  The BAR_NB_XPORT_COUNT parameter specifies the number of buffers for ON-Bar to use when communicating with INFORMIX-OnLine. The default is 10.

- BAR_XFER_BUF_SIZE

  The BAR_XFER_BUF_SIZE parameter is the size of each buffer specified in BAR_NB_XPORT_COUNT. The buffer size is actually set to the BAR_XFER_BUF_SIZE multiplied by the size of each page. The default is 31 when PAGESIZE is 2K. The default is 15 when PAGESIZE is 4K.

  Attention

  When we left the BAR_XFER_BUF_SIZE value set to 31, we received many messages (“WARNING: BAR_XFER_BUF_SIZE exceeded maximum allowed limit. Changing buffer size to 61440.”) in the /tmp/bar_act.log. The messages seemed to be informational. To eliminate the messages, we set the BAR_XFER_BUF_SIZE value to 4.

- ALARMPROGRAM

  The ALARMPROGRAM parameter is the full path name of a shell script called when a log full event occurs. The ALARMPROGRAM parameter allows logical logs to be automatically backed up to the ADSM storage manager. The default value for the ALARMPROGRAM parameter is the

$INFORMIXDIR/etc/log_full.sh shell script. If you do not want logical logs automatically backed up, set the ALARMPROGRAM parameter to $INFORMIXDIR/etc/no_log.sh, which is a shell script that prevents automatic backup of logical logs to the ADSM storage manager.

Attention

If the ALARMPROGRAM parameter is set to use the log_full.sh script, do not use the **onbar -l -c** command directly from the command line or through an automated ADSM server schedule. Back up logical logs either automatically or from a command. If you select the automatic method of logical log backup, set the ALARMPROGRAM parameter to reference the $INFORMIXDIR/etc/log_full.sh script or write your own similar script. If you decide to back up logical logs by command line or through the ADSM scheduler, set the ALARMPROGRAM parameter to reference the $INFORMIXDIR/etc/no_log.sh script or write your own similar script that ignores log change events. The log_full.sh script is the default.

If the ALARMPROGRAM parameter is set to $INFORMIXDIR/etc/log_full.sh and you issue an **onbar -l -c** command from the command line, a "WARNING: A log backup is already running. Can't start another." message is placed in the /tmp/bar_act.log. This was a point of confusion for us in testing until we understood that we should select only one method of logical log backup, not both. The warning message is displayed because onbar -l -c closes the current log so the alarmprogram is triggered. The alarmprogram tries to perform a second log backup, which results in the warning message being displayed. The message is not a "fatal error"; the logs are still safely backed up.

ON-Bar also stores information about INFORMIX-OnLine server instances, storage manager applications, XBSA versions, and backup objects and instances in catalog tables in the **sysutils** database. These tables are prefixed with *bar_*, and you can view them through the INFORMIX-OnLine dbaccess utility, using standard SQL commands.

ADSM Configuration Considerations: As we worked through our various scenarios, we discovered some unique configuration issues regarding the interaction of ON-Bar and ADSM. The considerations described in this section include dsm.sys configuration file changes, scheduling, backup copygroups, naming conventions for backup data, deleting objects from the ADSM database, errors in the /tmp/bar_act.log, and open issues.

- **dsm.sys** configuration file changes

  Only one change must be made in the ADSM dsm.sys client system configuration file for ON-Bar to interoperate with the ADSM X/OPEN API code. By default, ADSM client files are located in /usr/lpp/adsm/bin.

  As noted in 11.3.6.1, "ON-Bar and ADSM X/OPEN API Setup" on page 374, the password option for the client must be explicitly set to generate. The passwordaccess option, by default, is implicitly set to prompt. By implicitly set, we mean that the passwordaccess option is not present in the dsm.sys file. Figure 224 on page 380 is our dsm.sys file where you can see how we explicitly set the passwordaccess option to generate.

```
     SErvername            yellow
        COMMmethod         TCPip
        TCPPort            1500
        Passwordaccess     generate
```

Figure 224. Our dsm.sys File

- **Scheduling**

  To use the ADSM storage manager scheduling facilities with ON-Bar, you create a schedule and then associate an ON-Bar node with the schedule. Figure 225 shows the ADSM commands we used to create the schedule, and Figure 226 shows the command we used to associate an ON-Bar node with the schedule. Review Chapter 5, “Automating Database Backups” on page 69 for more details on ADSM scheduling.

```
     adsm> def sch redbook level0bu t=c desc="Level 0 Backup" \
     cont> act=command obj="su - informix -c 'onbar -w -L 0'" \
     cont> startd=12/12/96 startt=11:55:00

Output from "adsm> q sch redbook level0bu f=d"

Policy Domain Name     REDBOOK
Schedule Name          LEVEL0BU
Description            Level 0 Backup
Action
Options
Objects                su - informix -c "onbar -w -L 0"
Priority               5
Start Date/Time        12/12/96   11:55:00
Duration               1 Hour(s)
Period                 1 Day(s)
Day of Week            Any
Expiration
Last Update by         ADMINISTRATOR
Last Update            12/12/96   11:54:22
```

Figure 225. Create a Schedule in ADSM As an ADSM Administrator

Issue the command in Figure 226 to associate the ON-Bar node (bering) in policy domain redbook with the level0bu schedule.

```
     adsm> def assoc redbook level0bu bering
```

Figure 226. Associate an ON-Bar Node with the ADSM Schedule

- **Backup copygroups**

  Data sent from ON-Bar to the ADSM storage manager is considered backup data, not archive data. By default, data sent through the ON-Bar utility is stored in backup copy groups.

  When planning version and retention parameters for backup copy groups, certain unique facts about the ON-Bar utility are important to consider. First, INFORMIX-OnLine incremental backups are cumulative, unlike incremental

backups made with other utilities, including ADSM client incremental backups. As an example, a level-0 backup made on Sunday captures all database structure and data information. A level-1 backup made on Monday captures all changes made to the database structure and data information since the level-0 backup on Sunday. A level-1 backup on Tuesday also captures all changes made to the database structure and data information since the level-0 backup on Sunday.

Cumulative incrementals have several advantages and unique considerations. The cumulative nature of ON-Bar incremental backups results in a much quicker restoration of a failed dbspace or blobspace because fewer files must be referenced to re-create the environment. There is a significant time advantage in disaster recovery situations. Cumulative incrementals may require more disk storage, however, because they contain data since the level-0 backup, not the previous incremental backup.

You must address the usual ADSM considerations when defining backup copygroups: data restoration requirements, number of versions to retain, and what to do with the last copy of data. You may be tempted to decrease the number of versions retained because incremental backups are cumulative. Therefore, if you do this, clearly understand which point-in-time restoration requirements exist. Decreasing the number of versions of data retained may not be a good idea because of how the ON-Bar utility passes file names to the ADSM storage manager.

- **Naming conventions for backup data**

  When ON-Bar passes files to the ADSM storage manager, the data, by default, is stored on storage pool media associated with the backup copy group. As an example of naming conventions, we stored a level-0 backup on the ADSM storage manager and then several level-1 backups. In our test environment, we backed up data to a disk storage pool with a file name of /backup/00000009.BFS. Figure 227 on page 382 shows the output from a query of media contents.

```
Output from "adsm> q content /backup/00000009.BFS f=d"

Node Name:             BERING
Type:                  Bkup
Filespace Name:        /learn
Client Name for File:  /learn/rootdbs/ 0
Stored Size:           4.50M
Segment Number:        1/1
Cached Copy:           No

Node Name:             BERING
Type:                  Bkup
Filespace Name:        /learn
Client Name for File:  /learn/rootdbs/ 1
Stored Size:           525,262
Segment Number:        1/1
Cached Copy:           No

Node Name:             BERING
Type:                  Bkup
Filespace Name:        /learn
Client Name for File:  /learn/rootdbs/ 1
Stored Size:           722,062
Segment Number:        1/1
Cached Copy:           No

Node Name:             BERING
Type:                  Bkup
Filespace Name:        /learn
Client Name for File:  /learn/rootdbs/ 1
Stored Size:           754,862
Segment Number:        1/1
Cached Copy:           No
```

Figure 227. Naming Conventions for ON-Bar Files in the ADSM Storage Manager

If you study Figure 97 carefully, you may be surprised (or disturbed) to discover that all level-1 backups appear to use the same name, /learn/rootdbs/1. The increasing size of the level-1 backup files demonstrates the cumulative incremental backup method. We addressed the naming convention issue with developers from the IBM ADSM team and the INFORMIX-OnLine ON-Bar team and learned that the only unique identifier associated with each backed up file is a copy ID number. The copy ID number is not visible to an ADSM administrator, only a developer can see this unique field.

Also, the INFORMIX-OnLine database administrator and the ADSM administrator will not see the same information when querying their respective backup "catalog" tables. In the ADSM database, as shown in Figure 227, the ADSM administrator can see only those files named with the backup level used to create them. No date or time stamp is available. The INFORMIX-OnLine database administrator can query the *bar_* prefixed tables, also called **catalog tables**, in the INFORMIX-OnLine database, which are tables that are created and updated by ON-Bar for each backup or restore. The catalog tables clearly show a date and time stamp associated with each backup or restore. There is no human way to reconcile the ADSM database

and the ON-Bar catalog tables because the only unique field between them, the copy ID field, is externally “invisible” in ADSM but visible in the ON-Bar catalog. The copyid is in the bar_instance table, and the start and end times for backups and restores is in the bar_action table. The possibility of experiencing database synchronization problems between the two database tracking methods exists and, without a visible copy ID field, the probability of manual recovery from synchronization problems is limited. This is why backup copies of the INFORMIX-OnLine emergency boot file and full backups of the ADSM database server must be maintained on a regular basis. A strong point of concern over this issue has been raised with development personnel.

Attention

We discovered a manual (and rather crude) method of reconciling the INFORMIX-OnLine catalog and ADSM database. It involves using the undocumented and unsupported ADSM *show* command, which is typically used for diagnostic purposes. If you use the show command, you do so at your own risk. If information displayed is inaccurate or some unpleasant side effect should occur through use of the command, IBM is under no obligation to fix the problem.

As shown in Figure 228 on page 384, you can use the show command to see which INFORMIX-OnLine files on the ADSM server are inactive, as well as when they were inserted into the ADSM database (date and time stamp) and when they were deactivated from the ADSM database (date and time stamp).

To synchronize the INFORMIX-OnLine catalog and ADSM database, we recommend that you not run ADSM file expiration processing with the default time of every hour. Instead schedule file expiration processing as part of a script that would include:

1. Issuing the show command to get a list of inactive files
2. Running file expiration processing
3. Issuing the show command to get a list of files that can be used to compare with the list from the first show command (step 1).
4. Informing the INFORMIX-OnLine database administrator to delete the files that do not show up on the list from the second show command (step 3). You may be able to write a program to automate the INFORMIX-OnLine deletion process.

```
adsm> SHOW Versions bering learn

**************************************************************

Active, Inserted 10/31/1996 17:09:24

/learn : /learn/rootdbs/ 0 (MC: DEFAULT)
Active, Inserted 10/31/1996 18:18:28

/learn : /learn/rootdbs/ 1 (MC: DEFAULT)
Active, Inserted 10/31/1996 18:23:54

/learn : /learn/1/ 1 (MC: DEFAULT)
Inactive, Inserted 10/28/1996 20:14:24, Deactivated 10/31/1996 18:26:21

/learn : /learn/1/ 1 (MC: DEFAULT)
Inactive, Inserted 10/31/1996 18:26:21, Deactivated 10/31/1996 18:31:30

/learn : /learn/1/ 1 (MC: DEFAULT)
Inactive, Inserted 10/31/1996 18:31:30, Deactivated 10/31/1996 18:52:09

/learn : /learn/1/ 1 (MC: DEFAULT)
Inactive, Inserted 10/31/1996 18:52:09, Deactivated 10/31/1996 19:03:26
```

Figure 228. Using the Undocumented and Unsupported ADSM Show Command

Another consideration for copy group definitions depends on how far back you want point-in-time recovery. For example, in a simple environment, a full level-0 backup on Sunday coupled with six level-1 backups Monday through Saturday might be sufficient. If the copy group definition for the backup copy group set the Versions Exist parameter to 7, on Monday of the second week, the seventh level-1 backup is added to the backup copy group storage pool. On Tuesday of the second week, addition of the next level-1 backup means that the Versions Exist parameter has been exceeded, so the oldest level-1 backup, which is the level-1 backup from Monday of week 1, is deleted. At this point, you can recover to any point except the incremental backup point created on Monday of week 1. Although somewhat tedious to work through, these are the kinds of detailed planning scenarios that must be defined with the customer to ensure that restoration requirements match ADSM storage manager parameter definitions.

- **Deleting objects from the ADSM database**

  ON-Bar files in ADSM appear to use the same names for each level-x backup type. The objects are removed from the ADSM server through normal ADSM copy group parameters for version retention coupled with expiration processing. This creates the synchronization problems between the ADSM database and ON-Bar catalog tables. ON-Bar does not issue any delete object calls through the X/OPEN API. All file expiration processing is left to the ADSM storage manager.

  To view files backed up to ADSM, you can query specific volumes where data is stored. A list of files backed up to ADSM is also stored in the ON-Bar utility catalog files which are stored in the **sysutils** database table. The catalog tables that store backup information from ON-Bar begin with the prefix *bar_*. In our testing of the current ON-Bar utility in INFORMIX-OnLine

Version 7.21 and the ADSM client code Version 2.1.0.5, there is no cross-updating of backup records by either product.

If files are deleted from the ADSM storage manager database through expiration or some other action, the catalog tables in the ON-Bar utility still indicate that the backups exist. Similarly, if an INFORMIX-OnLine database administrator deletes records from the *bar_* prefixed tables, the ADSM storage manager administrator will see indications that the files exist because the ON-Bar utility does not send delete object calls through the ADSM X/OPEN API. Until the cross-updating issue is addressed, it is critical that ADSM storage managers and INFORMIX-OnLine database administrators clearly and accurately communicate backup and restore requirements and manually back up and maintain their respective databases until a data synchronization feature is available from one of the vendors.

- **Errors in the /tmp/bar_act.log**

  In this section we document two sets of errors we saw with some frequency in the ON-Bar bar_act.log log file. Neither /tmp/bar_act.log error message is a true indicator of a failure condition.

  The first set of errors occurs on a full, cold restore of the database and on a logical log recovery for a database or dbspace or blobspace. When you attempt either of these recoveries, there is frequently an “Invalid XBSA function call sequence” message. The error is caused by improper passing and handling of the unique copy ID associated with the salvaged logical log. It does not appear to prevent or corrupt the restore process, however. Development is investigating the error.

  The message, “An unspecified XBSA error has occurred: 96,” generally means that connection from the ON-Bar utility to the ADSM storage manager cannot be established. We saw the error when the ADSM storage manager was down. To resolve this problem, we read through the bar_act.log and adsm_trace files in the /tmp directory. You should also check that the ADSM dsm.opt and dsm.sys files are correct.

  Figure 229 on page 386 shows the second set of errors in the /tmp/bar_act.log..

```
1996-10-25 08:08:18 26550  15346 onbar -b -L 0
1996-10-25 08:08:21 26550  15346 Begin backup logical log 1.
1996-10-25 08:08:22 26550  15346 Successfully connected to Storage Manager.
1996-10-25 08:09:11 26550  15346 Completed backup logical log 1.
1996-10-25 08:09:11 26550  15346 Begin backup logical log 2.
1996-10-25 08:10:01 26550  15346 Completed backup logical log 2.
1996-10-25 08:10:02 26550  15346 Begin backup logical log 3.
1996-10-25 08:10:51 26550  15346 Completed backup logical log 3.
1996-10-25 08:10:56 26550  15346 Begin level 0 backup rootdbs.
1996-10-25 08:10:57 26550  15346 Successfully connected to Storage Manager.
1996-10-25 08:13:08 26550  15346 Completed level 0 backup rootdbs.
1996-10-25 08:10:56 26550  15346 Begin level 0 backup mydbspace.
1996-10-25 08:10:57 26550  15346 Successfully connected to Storage Manager.
1996-10-25 08:13:08 26550  15346 Completed level 0 backup mydbspace.
1996-10-25 08:13:08 26550  15346 Invalid XBSA function call sequence.
1996-10-25 08:13:08 26550  15346 SQL -329 Database not found or no
                                 system permission.
1996-10-25 08:13:08 26550  15346 ISAM -155 ISAM error: Primary and
                                 Mirror chunks are bad
1996-10-25 08:13:08 26550  15346 ON-Bar is waiting for the OnLine
                                 server to exit fast recovery mode
```

Figure 229. Errors in /tmp/bar_act.log after Cold Database Restore

We also received the errors when we tried to do a point-in-time restore using the **onbar -r -t 1996-10-25 08:13:00** command. By the time the errors register in the /tmp/bar_act.log, the onbar process is completed. We learned that the errors indicate that the INFORMIX-OnLine database server has not yet changed into a multiuser or quiescent mode. The errors do not indicate that the restore of the database has failed, but rather that the database is going through a fast recovery process. We verified that the database restored properly by issuing **onstat** commands from the informix userid.

11.3.6.2 Full Online Database Server Level-0 Backup

As discussed in 11.2.4, “ON-Bar” on page 332, ON-Bar provides excellent facilities for backing up INFORMIX-OnLine databases. It is possible to perform a backup of an INFORMIX-OnLine database while the database is online and available.

In our test, we demonstrated a weekly level-0 backup, that is, a baseline backup of the database. A level-0 backup contains a copy of all *pages* that contain data for every dbspace and blobspace. All of the *pages* are needed to restore the database should the machine’s disk be completely destroyed. We invoked the level-0 backup from the command line on the machine where the INFORMIX-OnLine database was installed. Figure 230 shows the syntax.

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

Figure 230. Level-0 Command Line Backup

To check on the progress of the backup, we used the UNIX *tail* command to view the ON-Bar activity log. The location of the log is determined by the BAR_ACT_LOG parameter in the onconfig file used to start the database. Figure 231 on page 387 shows the *tail* command we used.

```
tail -f /tmp/bar_act.log
```

Figure 231. tail Command Syntax

Output in the /tmp/bar_act.log from a successful level-0 backup looks similar to that shown in Figure 232.

```
1996-10-25 08:08:18 26550  15346 onbar -b -L 0
1996-10-25 08:08:21 26550  15346 Begin backup logical log 1.
1996-10-25 08:08:22 26550  15346 Successfully connected to Storage Manager.
1996-10-25 08:09:11 26550  15346 Completed backup logical log 1.
1996-10-25 08:09:11 26550  15346 Begin backup logical log 2.
1996-10-25 08:10:01 26550  15346 Completed backup logical log 2.
1996-10-25 08:10:02 26550  15346 Begin backup logical log 3.
1996-10-25 08:10:51 26550  15346 Completed backup logical log 3.
1996-10-25 08:10:56 26550  15346 Begin level 0 backup rootdbs.
1996-10-25 08:10:57 26550  15346 Successfully connected to Storage Manager.
1996-10-25 08:13:08 26550  15346 Completed level 0 backup rootdbs.
1996-10-25 08:10:56 26550  15346 Begin level 0 backup mydbspace.
1996-10-25 08:10:57 26550  15346 Successfully connected to Storage Manager.
1996-10-25 08:13:08 26550  15346 Completed level 0 backup mydbspace.
```

Figure 232. /tmp/bar_act.log Output after a Level-0 Backup

The logical logs are backed up first during a level-0 backup and then each of the dbspaces is backed up. Every INFORMIX-OnLine database has a rootdbs instance. In our example, we created a second dbspacename, *mydbspace*. This dbspace was also backed up during the level-0 backup.

11.3.6.3 Full Online Database Server Level-1 Backup

Level-1 backups capture changes to the database since the last level-0 backup. A level-1 backup takes less time to create than a level-0 backup because only data changes since the last level-0 backup are sent to the ADSM storage manager. Because level-1 backups are incremental from the last level-0 backup, we recommend that level-0 backups be made at least once a week to decrease the number of level-1 backups that must be retained.

The INFORMIX-OnLine database also allows for the creation of level-2 backups. These backups store changes to the database since the creation of the last level-1 backup. We do not provide examples of level-2 backups in our scenarios as these backups behave like level-1 backups.

We invoked the level-1 backup from the command line on the machine where the INFORMIX-OnLine database was installed. Figure 233 shows the syntax.

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

Figure 233. Level-1 Command Line Backup

We used the UNIX *tail* command to view the On-Bar activity log to check on the progress of the backup. Figure 234 on page 388 shows the output from a successful level-1 backup.

```
1996-10-25 09:48:42 26596  15346 onbar -b -L 1
1996-10-25 09:48:47 26596  15346 Begin level 1 backup rootdbs.
1996-10-25 09:48:48 26596  15346 Successfully connected to Storage Manager.
1996-10-25 09:49:36 26596  15346 Completed level 1 backup rootdbs.
1996-10-25 09:48:47 26596  15346 Begin level 1 backup mydbspace.
1996-10-25 09:48:48 26596  15346 Successfully connected to Storage Manager.
1996-10-25 09:49:36 26596  15346 Completed level 1 backup mydbspace.
```

Figure 234. /tmp/bar_act.log Output after a Level-1 Backup

11.3.6.4 Online Backup of a Database Space

We describe a physical back up of a dbspace called *mydbspace*. The /tmp/filespace.file contains a single line, mydbspace, to identify the dbspacename to be backed up. Figure 235 shows the contents of /tmp/filespace.file. The -f flag on the onbar command is useful if you have a large number of dbspaces to back up and do not want to type all of them into the system. If the ADSM storage manager scheduler is used to invoke this command, the file referenced by the -f flag must exist in the proper directory on the client machine with proper execute permissions assigned.

```
mydbspace
```

Figure 235. Contents of /tmp/filespace.file

We issued the onbar command with the -f flag as shown in Figure 236.

```
# onbar -b -f /tmp/filespace.file
```

Figure 236. Physical Dbspace Backup

Figure 237 shows the output from a successful backup.

```
1996-10-25 13:48:42 26596  15346 onbar -b -f /tmp/filespace.file
1996-10-25 13:48:47 26596  15346 Begin level 0 backup mydbspace
1996-10-25 09:48:48 26596  15346 Successfully connected to Storage Manager.
1996-10-25 09:49:36 26596  15346 Completed level 0 backup mydbspace.
```

Figure 237. /tmp/bar_act.log Output after a Physical Dbspace Backup

11.3.6.5 Full Restore to the Current Point-in-Time

A cold restore (and an optional point-in-time recovery) is necessary if the physical disk on which the database resides fails. If the physical disk fails, basic recovery tasks, such as recovery of the operating system, and reconfiguration of operating system utilities, such as TCP/IP, must be accomplished. Recovery of the basic operating system includes ADSM client software. The minimum set of customized ADSM client files that must be restored includes:

- /usr/lpp/adsm/bin/dsm.opt
- /usr/lpp/adsm/bin/dsm.sys
- /usr/lpp/adsm/api/bin/libXApi.a relinked to /usr/lib/ibsad001.a

Recovery also means that INFORMIX-OnLine database executables and critical, customized INFORMIX-OnLine files are in place before data can be pulled from the ADSM storage manager. The customized INFORMIX-OnLine files that must be restored before the ON-Bar utility can restore the database to the current point-in-time include the:

- Emergency boot file, $INFORMIXDIR/etc/ixbar.servernum
- onconfig file, $INFORMIXDIR/etc/onconfig
- Locator file, $INFORMIXDIR/etc/oncfg_$INFORMIXSERVER.SERVERNUM
- sqlhosts file, $INFORMIXDIR/etc/sqlhosts

11.3.6.6 Cold Restore of the Database Server

A cold restore of the database server implies that the database is not available during the restore process. The ON-Bar utility allows for several types of cold, offline restores of the entire database. The three types of restoration we consider are restore to the current point-in-time, restore to a specific point-in-time, and restore to a specific logical log.

To restore a failed database, the database must be in an offline condition. To check the condition of your database, issue an **onstat -** command and verify that the top line of output does not show the database in either a Quiescent or OnLine condition. If the onstat - command returns a message that shared memory for your database is not initialized, the database is down.

Restore to the Current Point-in-Time: To invoke the full restore, issue the command shown in Figure 238.

```
onbar -r
```

Figure 238. ON-Bar Command Syntax for a Cold Restore of an Offline Database

Figure 239 on page 390 shows the output after the restore completes.

```
1996-10-25 09:48:42 26596 16523  onbar -r
1996-10-25 10:00:00 27525 16523  Successfully connected to Storage Manager.
1996-10-25 10:00:00 27525 16523  Begin salvage for log 3.
1996-10-25 10:00:00 27525 16523  Complete salvage of logical log 3.
1996-10-25 10:00:00 27525 16523  Successfully connected to Storage Manger.
1996-10-25 10:00:00 27525 16523  Begin cold level 0 restore rootdbs.
1996-10-25 10:00:00 27525 16523  Completed cold level 0 restore rootdbs.
1996-10-25 10:00:00 27525 16523  Process 34036  36050 successfully forked.
1996-10-25 10:00:00 27525 16523  Process 34546  36050 successfully forked.
1996-10-25 10:00:00 27525 16523  Successfully connected to Storage Manger.
1996-10-25 10:00:00 27525 16523  Begin cold level 0 restore mydbspace.
1996-10-25 10:00:00 27525 16523  Process 34036  36050 completed.
1996-10-25 10:00:00 27525 16523  Process 34546  36050 completed.
1996-10-25 10:00:00 27525 16523  Begin cold level 0 restore loglv.
1996-10-25 10:00:00 27525 16523  Completed cold level 0 restore loglv.
1996-10-25 10:00:00 27525 16523  Process 34546  36050 completed.
1996-10-25 10:00:00 27525 16523  Successfully connected to Storage Manager.
1996-10-25 10:00:00 27525 16523  Begin restore logical log 1.
1996-10-25 10:00:00 27525 16523  Completed restore logical log 1.
1996-10-25 10:00:00 27525 16523  Begin restore logical log 2.
1996-10-25 10:00:00 27525 16523  Completed restore logical log 2.
```

Figure 239. /tmp/bar_act.log Output after a Cold Offline Restore

Restore to a Specific Point-in-Time: Another type of restoration which requires the database to be in an offline condition is an offline restoration to a specific point-in-time. The ON-Bar utility provides several nice features in its point-in-time recovery command. First, the ON-Bar utility allows you to enter any time in the proper format and ON-Bar handles the gathering and restoration of all required files. For example, if you want to restore the database to the state which existed at 17:00:00 hours on last Sunday evening, simply issue the command in Figure 240.

```
onbar -r -t "1996-10-31 17:00:00"
```

Figure 240. ON-Bar Command Syntax for Cold Point-in-Time Restore

Note: Time is specified in military format. The date and time must conform to the ANSI-style date and time format.

The ON-Bar point-in-time command is easy to use because you do not have to worry about uncommitted transactions that may have existed at 17:00:00 hours last Sunday evening. When the ON-Bar utility completes the requested restore process, it automatically does a rollback of transactions listed as uncommitted in the transaction log, as a final step. The restored database is ensured to be in a stable condition after the restore completes.

If you want to be more specific in selecting a time for restoration, read the INFORMIX-OnLine emergency boot file which lists all backup and recovery activity forwarded through ON-Bar to the ADSM storage manager. Figure 241 on page 391 is a copy of our emergency boot file, $INFORMIXDIR/etc/ixbar.1.

```
learn rootdbs R 1 1 0 0 63489 1996-10-31 14:18:01 1
learn rootdbs R 0 2 1 0 63490 1996-10-31 15:45:01 1
learn rootdbs R 0 3 1 0 63491 1996-10-31 16:59:59 1
```

Figure 241. INFORMIX-OnLine Emergency Boot File Contents

Note as you read through this example that our restoration time, 1996-10-31 17:00, is slightly more recent than the actual backup creation time.

You would perform a point-in-time restore to restore a specific backup copy (which is what our example above shows) or to restore to just before an error occurred, for example, if someone accidentally deleted an important table. In the second case, you must get the time the table was deleted and restore to a few seconds before that time. It does not matter when the backups were performed. ON-Bar gets the latest backup copy before the point in time you specified, restores the backup copy, and rolls forward the logs to that time and stops.

As shown in Figure 242, we tail the output of the bar_act.log file to view restore progress.

```
1996-10-25 10:00:00 27525 16523  onbar -r -t 1996-10-27 17:00
1996-10-25 10:00:00 27525 16523  Successfully connected to Storage Manager.
1996-10-25 10:00:00 27525 16523  Begin salvage for log 3.
1996-10-25 10:00:00 27525 16523  Complete salvage of logical log 3.
1996-10-25 10:00:00 27525 16523  Successfully connected to Storage Manger.
1996-10-25 10:00:00 27525 16523  Begin cold level 0 restore rootdbs.
1996-10-25 10:00:00 27525 16523  Completed cold level 0 restore rootdbs.
1996-10-25 10:00:00 27525 16523  Process 34036  36050 successfully forked.
1996-10-25 10:00:00 27525 16523  Process 34546  36050 successfully forked.
1996-10-25 10:00:00 27525 16523  Successfully connected to Storage Manger.
1996-10-25 10:00:00 27525 16523  Begin cold level 0 restore mydbspace.
1996-10-25 10:00:00 27525 16523  Process 34036  36050 completed.
1996-10-25 10:00:00 27525 16523  Process 34546  36050 completed.
1996-10-25 10:00:00 27525 16523  Begin cold level 0 restore loglv.
1996-10-25 10:00:00 27525 16523  Completed cold level 0 restore loglv.
1996-10-25 10:00:00 27525 16523  Process 34546  36050 completed.
1996-10-25 10:00:00 27525 16523  Successfully connected to Storage Manager.
1996-10-25 10:00:00 27525 16523  Begin restore logical log 1.
1996-10-25 10:00:00 27525 16523  Completed restore logical log 1.
1996-10-25 10:00:00 27525 16523  Begin restore logical log 2.
1996-10-25 10:00:00 27525 16523  Completed restore logical log 2.
```

Figure 242. /tmp/bar_act.log Output after Cold Point-in-Time Restore

At the conclusion of this test, the INFORMIX-OnLine database is in a quiescent state. We used the INFORMIX-OnLine onmonitor utility to bring the database back online. You may want to check the consistency of the database before you bring it back online and make it available to users. See the *INFORMIX-OnLine Dynamic Server Administrator's Guide* for how to use *oncheck* to check your database.

We used the dbaccess utility from the INFORMIX-OnLine database to prove that the dbspace, mydbspace, existed and that mytable existed within the dbspace.

Remember for offline restore, you **must** restore all critical media, that is, the root dbspace and any dbspaces with a physical or logical log.

11.3.6.7 Warm Restore of an Instance Using ON-Bar and ADSM

The ability to restore specific dbspaces or blobspaces while the INFORMIX-OnLine database is online is a major advantage of the ON-Bar utility. To use our previous scenarios again, if the root dbspace on our system happened to be online and the dbspace, mydbspace, was physically corrupted (for example, the hard disk crashed), it is possible to restore the single dbspace without taking the INFORMIX-OnLine database away from other users. To perform a warm restore, we issued the command in Figure 243, while the remainder of the INFORMIX-OnLine database, such as the root dbspace and other dbspaces, remain active. Remember that a warm restore requires that the database be rolled forward until the end of the log. Point-in-time restores are always cold restores.

```
onbar -r mydbspace
```

Figure 243. ON-Bar Command Syntax for a Warm Restore of a Specific Dbspace

We could not test the warm restore scenario because we could not crash the mydbspace dbspace.

Our scenarios demonstrate how to use the INFORMIX-OnLine database ON-Bar utilities to perform both cold (offline) and warm (online) restoration of database data. The INFORMIX-OnLine database manuals also discuss a mixed restore method. A mixed restore is an offline recovery of critical dbspaces such as the root dbspace, so that the database can be brought online, and a warm restore of noncritical dbspaces. Combining any of our cold recovery scenarios with the warm dbspace restore scenario mimics the INFORMIX-OnLine mixed restore scenario.

11.3.7 Using ON-Bar to Automatically Archive Logical Logs

To protect your INFORMIX-OnLine system as much as possible, we recommend that you use an automatic process to back up your logical logs. Automatic archiving logical logs is discussed in 11.1.2, “Logical Logs” on page 328 and 11.3.6, “Full ′Online′ Backup Using ON-Bar and ADSM” on page 373. In 11.3.6, “Full ′Online′ Backup Using ON-Bar and ADSM” on page 373 we document that logical logs can be saved automatically if you use the ALARMPROGRAM parameter in the onconfig file *or* the **onbar -l** command from the command line. If you select both methods, warning messages are output in the /tmp/bar_act.log file. If you choose to back up logical log files manually or through the ADSM storage manager scheduler, be aware that other types of ON-Bar log backups will produce the warning messages. The log backups do not fail; the logs are still safely backed up.

Set the ALARMPROGRAM parameter in the onconfig file to $INFORMIXDIR/etc/log_full.sh to invoke the automatic logical log backup. If you want to back up logical logs by hand or through the ADSM storage manager scheduler, change the ALARMPROGRAM parameter in the onconfig file to

$INFORMIXDIR/etc/no_log.sh. Changes to this parameter in the onconfig file require that the online server be stopped and restarted to invoke the change. After you change this parameter so that log backup will not automatically start, issue the following command to backup all logs:

```
onbar -l
```

Figure 244. ON-Bar Logical Log Backup Command Syntax

After you issue the onbar -l command, you can issue an **onstat -l** command (as shown in Figure 245), to determine which logs have been marked as having been backed up. A *B* will be displayed in the flags field.

```
INFORMIX-OnLine Version 7.21.UC1A9 -- On-Line -- Up 00:06:44 -- 9344 Kb


address    number   flags   uniqid   begin    size   used   %used
30047d90     1     U---B--  1        100139    125    125   100.00
30047dac     2     U---B--  2        1001b6    125    125   100.00
30047dc8     3     U-C---L  3        1002b0    125    125   004.25
30047de4     4     F------  4        10032d    125    125   000.00
```

Figure 245. Output from onstat -l Command after Logical Log Backup

To back up and close the current log so that the INFORMIX-OnLine database can use a new log, issue the ON-Bar command in Figure 246.

```
onbar -l -c
```

Figure 246. ON-Bar Logical Log Backup Command Syntax

Figure 247 shows the results from another **onstat -l** command.

```
INFORMIX-OnLine Version 7.21.UC1A9 -- On-Line -- Up 00:06:44 -- 9344 Kb

.....

address    number   flags   uniqid   begin    size   used   %used
30047d90     1     U---B--  1        100139    125    125   100.00
30047dac     2     U---B--  2        1001b6    125    125   100.00
30047dc8     3     U---B--  3        1002b0    125    125   100.00
30047de4     4     U--C--L  4        10032d    125    125   001.00
```

Figure 247. Output from onstat -l Command after Another Logical Log Backup

11.3.8 Partial Online Backup (Raw) Using dbexport and ADSM

This example shows how to back up two databases (no logs) using the INFORMIX-OnLine export utility, dbexport. Although the database server can be in online mode, dbexport obtains exclusive locks on the tables, so user access to those tables is not allowed. Remember, dbexport is really a data migration utility; the export is not coordinated with the information stored in logical log

files, and it does not save a copy of system overhead information important to INFORMIX-OnLine. Thus inconsistencies in your database could result during the dbimport process.

With dbexport you export at the database level: one or up to all of your databases that comprise an INFORMIX-OnLine database server. In our example we use dbexport to export two databases, inf5testdb01 and inf5testdb02, to the /home/informix5 directory:

```
dbexport -o /home/informix5 inf5testdb01
dbexport -o /home/informix5 inf5testdb02
```

The dbexport created the directories shown in Figure 248.

```
# cd /home/informix5
# ls -al
total 32
drwxr-xr-x   4 informix informix     512 May  5 17:17 .
drwxr-xr-x  11 bin      bin          512 May  4 11:41 ..
drwxr-x---   2 informix informix     512 May  5 17:15 inf5testdb01.exp
drwxr-x---   2 informix informix     512 May  5 17:17 inf5testdb02.exp
#
# cd inf5testdb01.exp
# ls -al
total 3224
drwxr-x---   2 informix informix     512 May  5 17:15 .
drwxr-xr-x   4 informix informix     512 May  5 17:17 ..
-rw-r--r--   1 informix informix 3398544 May  5 17:15 emp____100.unl
-rw-r--r--   1 informix informix     746 May  5 17:15 inf5testdb01.sql
#
# cd ../inf5testdb02.exp
# ls -al
total 3224
drwxr-x---   2 informix informix     512 May  5 17:15 .
drwxr-xr-x   4 informix informix     512 May  5 17:17 ..
-rw-r--r--   1 informix informix 4160000 May  5 17:15 emp2___100.unl
-rw-r--r--   1 informix informix     746 May  5 17:15 inf5testdb02.sql
#
```

Figure 248. Directories Created by INFORMIX-OnLine dbexport

Dbexport creates a specific directory for each database and a specific file for each table you export. This can result in lots of directories and files, depending on how many databases or tables your database server manages. Now we use the ADSM backup/archive GUI client to back up these directories and files. As shown in Figure 249 on page 395, to back up the contents of the /home/informix5 directory with all of its subdirectories, select the /home file system as the start directory from the ADSM File System Information window.

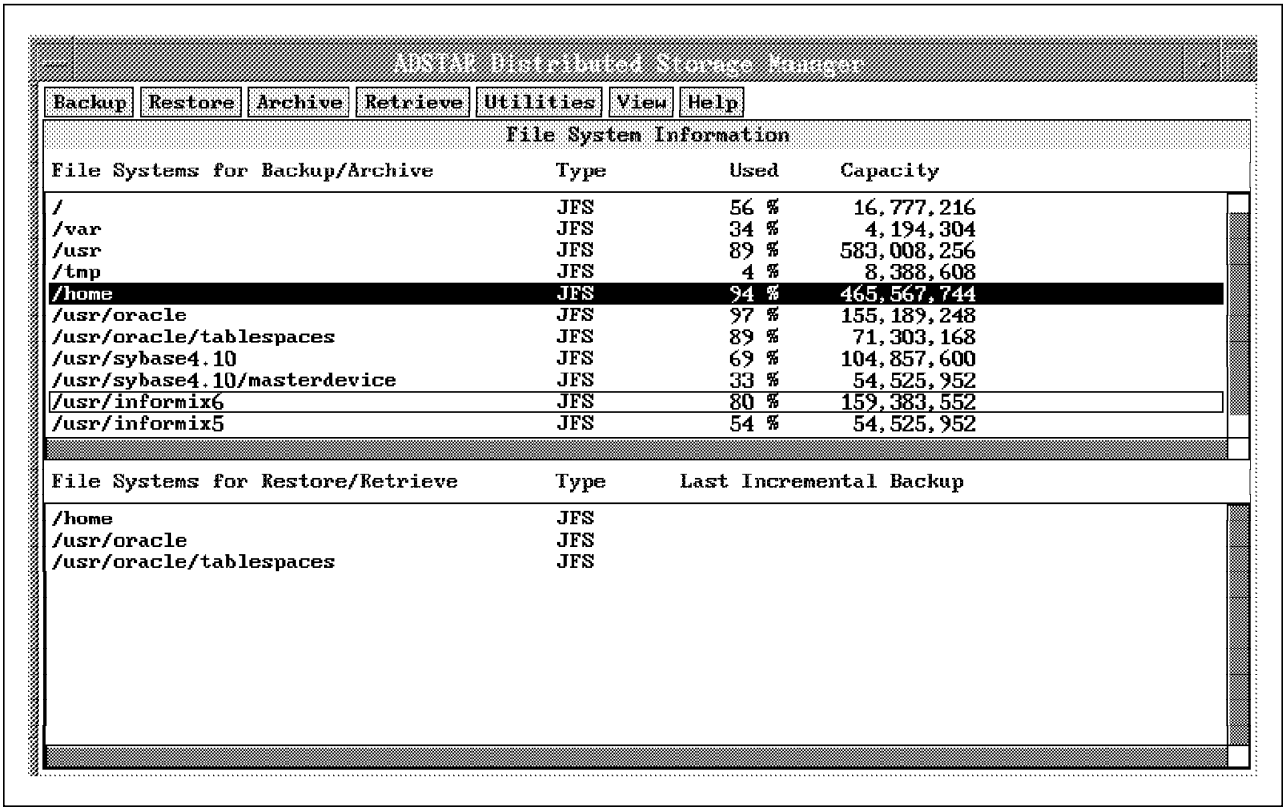


Figure 249. Select the INFORMIX-OnLine Directory for ADSM Backup

Click on Backup and select the Backup by directory tree option from the pull-down (not shown). Then select the /home/informix5/inf5testdb01.exp directory and its files from the Backup by Directory Tree window (see Figure 250 on page 396).

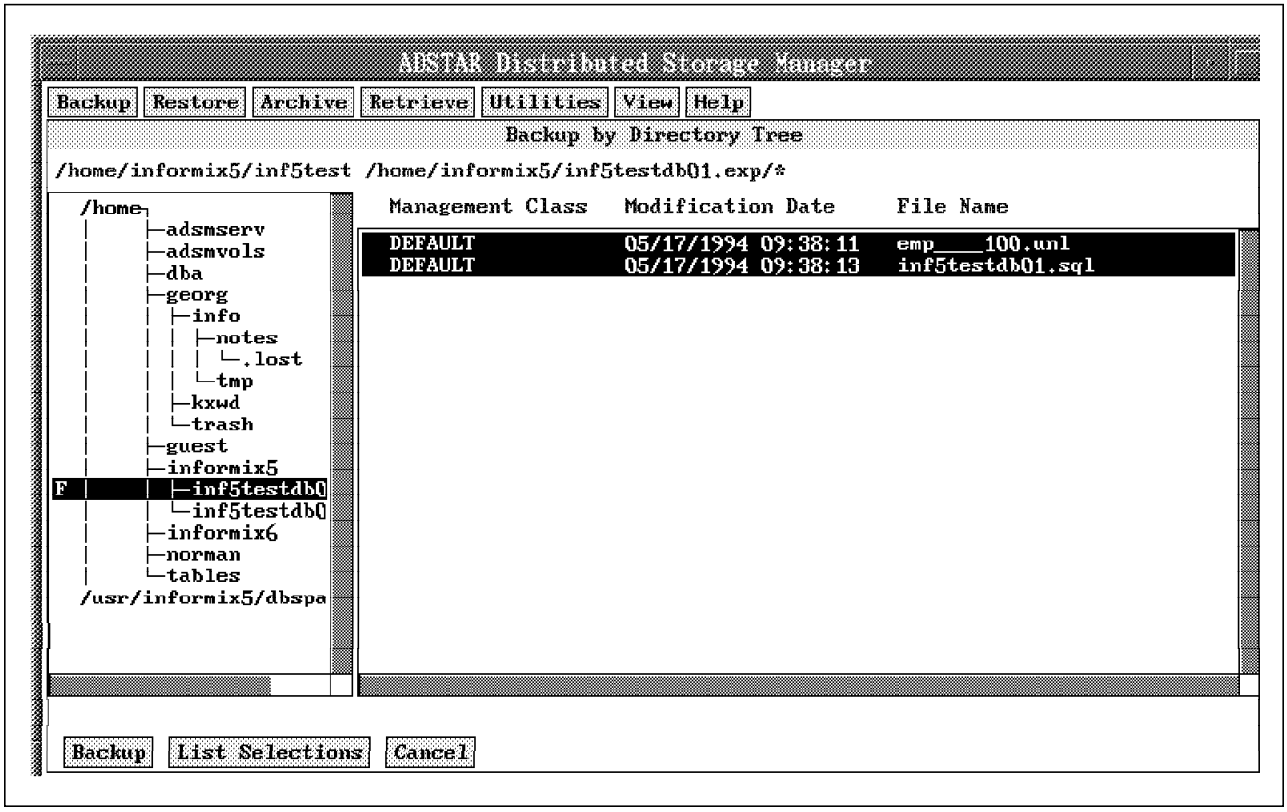


Figure 250. Select the Specific INFORMIX-OnLine Files for ADSM Backup

Click on Backup and the Backup Status window appears (see Figure 251 on page 397).

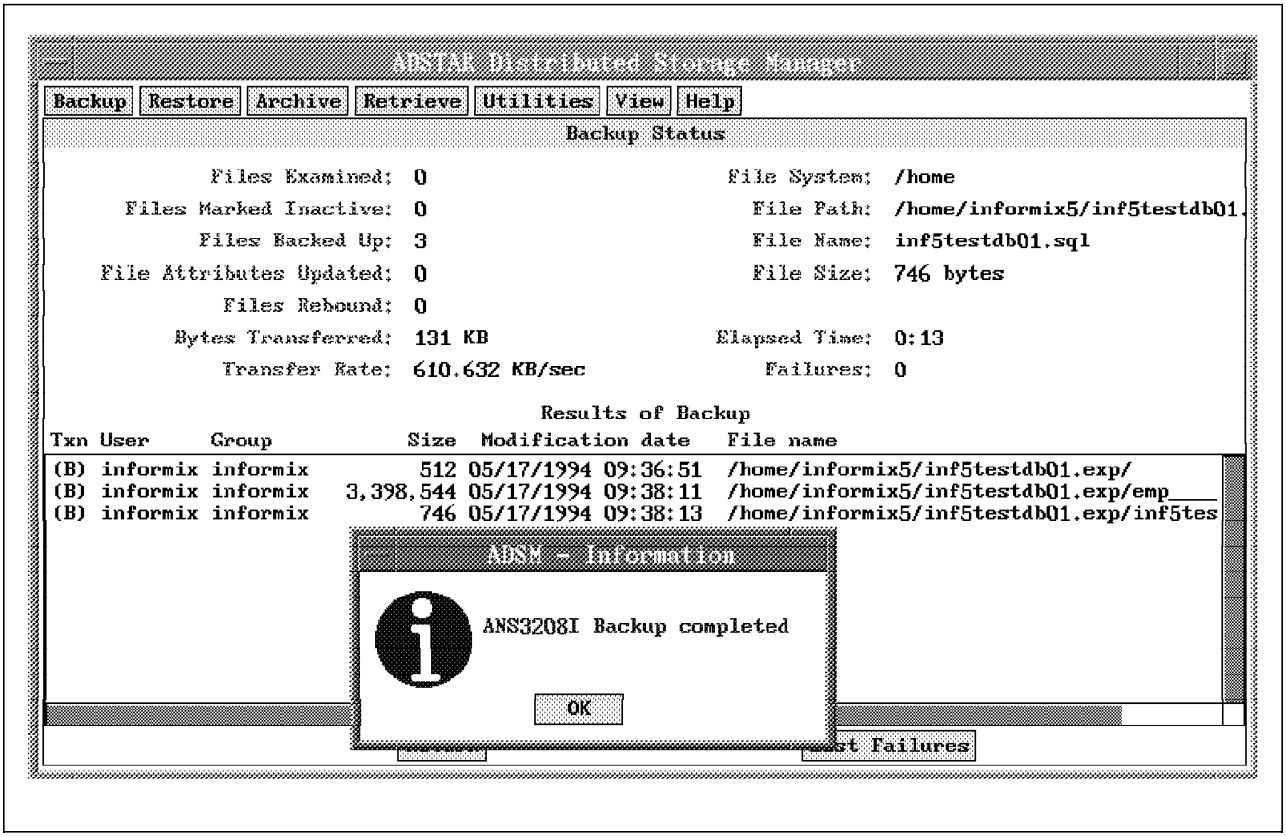


Figure 251. ADSM Backup of INFORMIX-OnLine Is Complete

Now repeat this backup procedure for the other subdirectory, /home/informix5/inf5testdb02.exp.

To restore the database, first restore the files from ADSM; then use the INFORMIX-OnLine dbimport utility to recreate the database. As shown in Figure 252 on page 398, select the /home file system from the ADSM File System Information window.

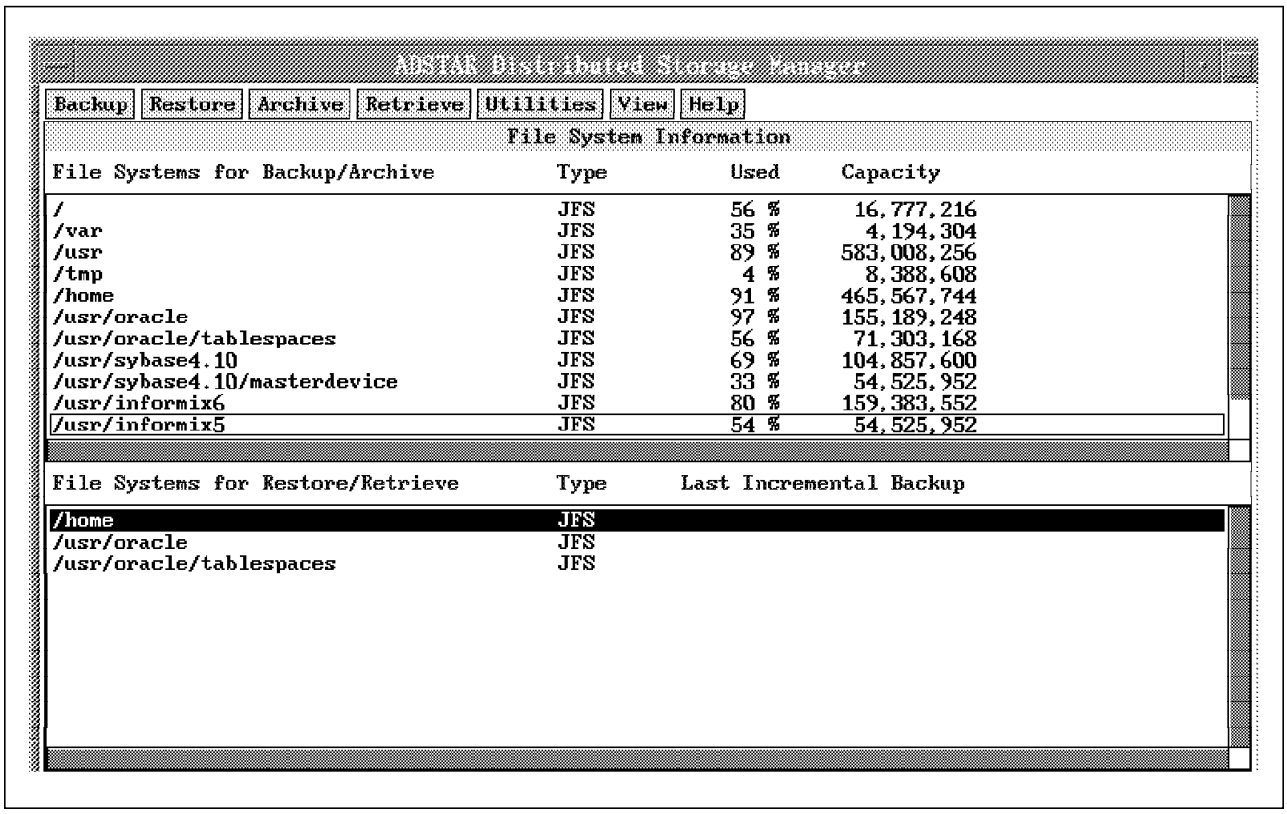


Figure 252. Select the INFORMIX-OnLine File System You Want to Restore with ADSM

Then click on Restore and select the Restore by directory tree option from the pull-down (not shown). On the Restore by Directory Tree window, select the appropriate directory from the left side, and the files will appear on the right side (see Figure 253 on page 399). Select the files you want to restore (in this case, emp____100.unl and inf5testdb01.sql).

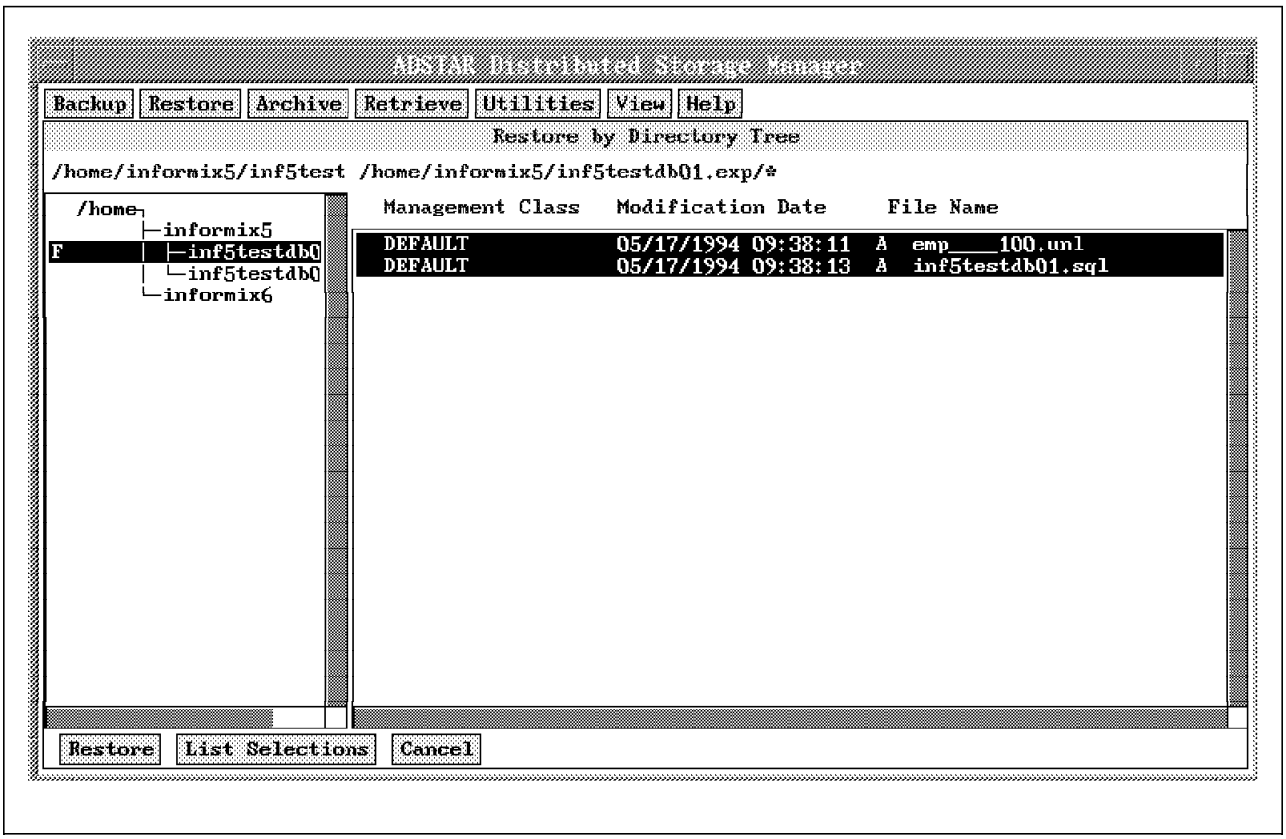


Figure 253. Select INFORMIX-OnLine Files for ADSM Restore

Click on Restore and the Restore/Retrieve Parameters window appears (see Figure 254 on page 400).

To demonstrate the flexibility of the ADSM backup/archive client, we show an example of how files and directories can be restored to another destination path.

We assume that there is not enough space in the /home file system to restore the files to their original destination in /home/informix5, so we restore the files to the /usr/inf5testdb01.exp and /usr/inf5testdb02.exp directories. Please note that the directories do not need to exist; ADSM creates them if they do not already exist.

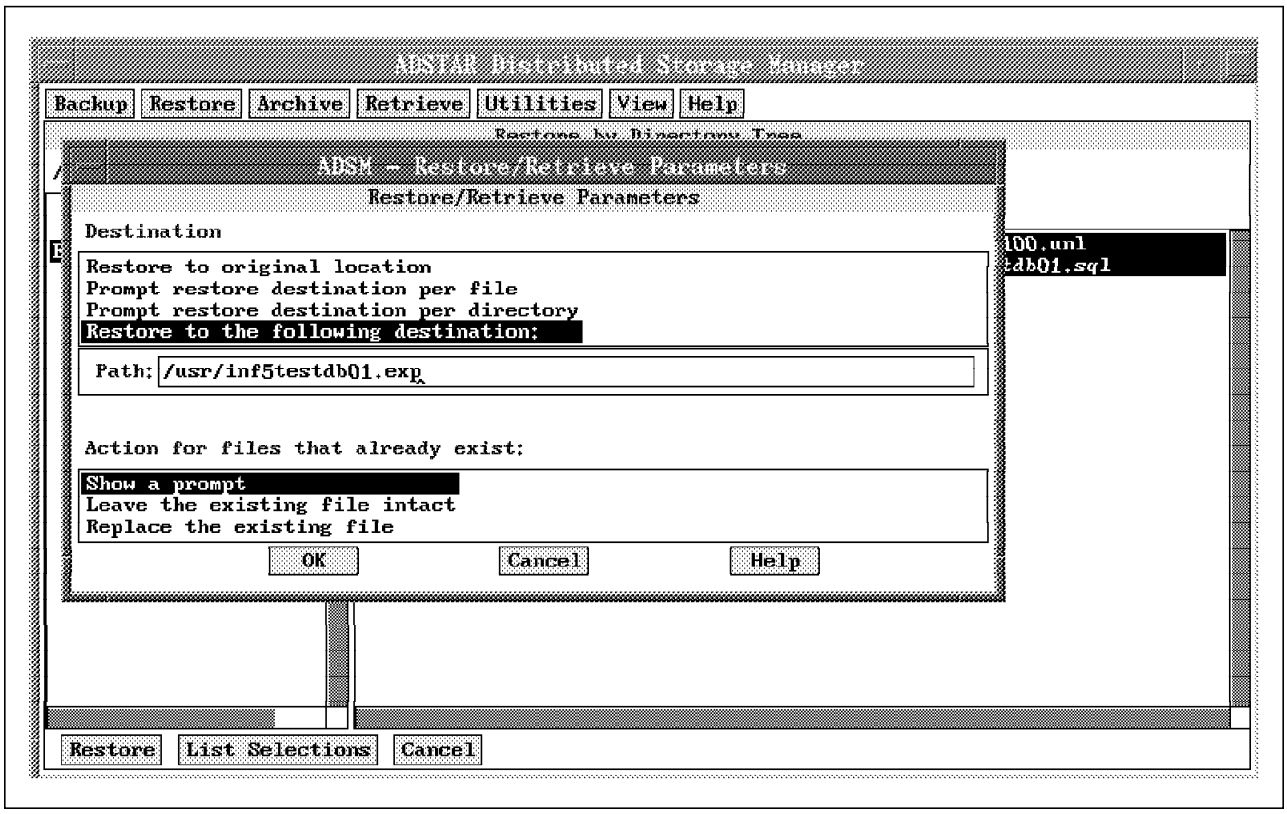


Figure 254. Enter the INFORMIX-OnLine Restore Destination

Click on OK to get to the Restore Status window (see Figure 255).

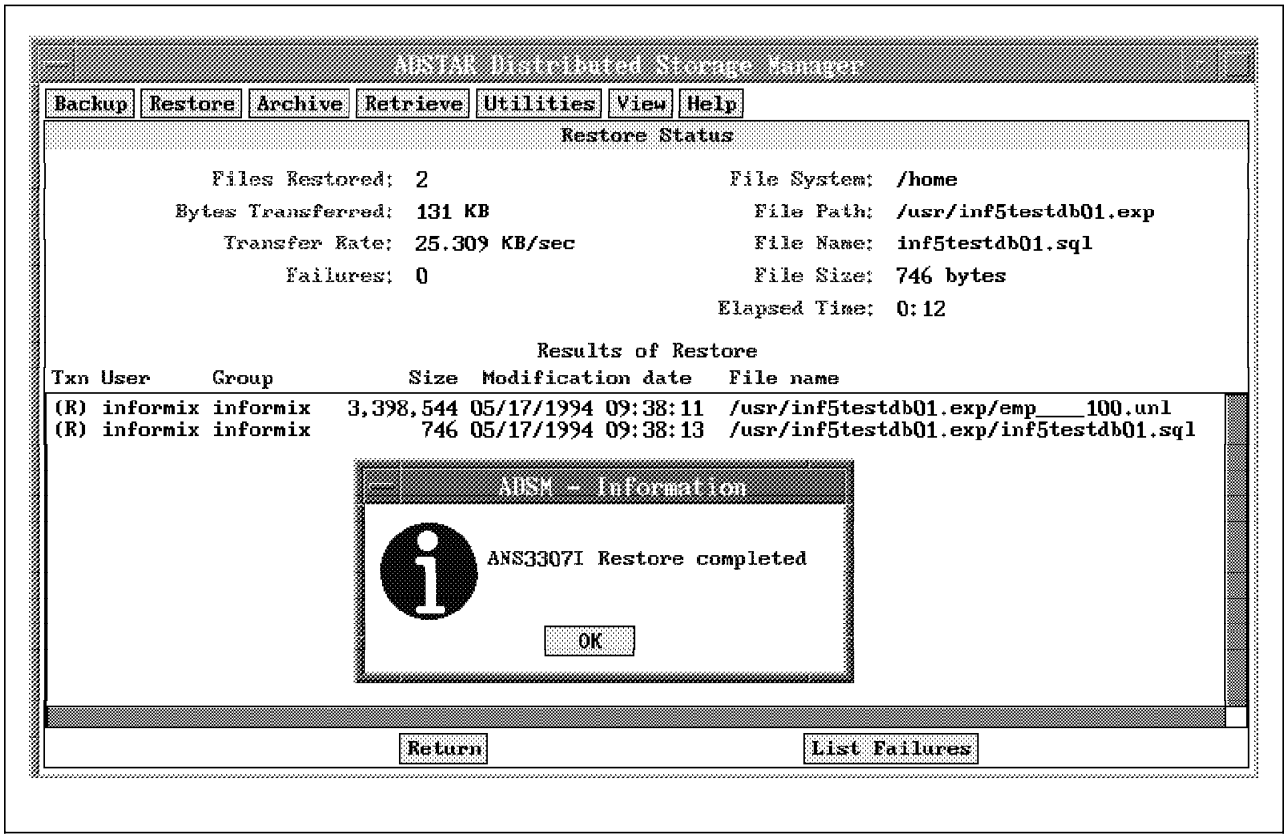


Figure 255. ADSM Restore of INFORMIX-OnLine Is Complete

To restore the second database, inf5testdb02, restore the files in the other directory, /home/inf5testdb02.exp, repeating the previous ADSM restore steps.

Now that we successfully restored the data from ADSM, we need to use the INFORMIX-OnLine dbimport utility to re-create the database. Dbimport re-creates the tables of a database and the database itself from its SQL schema file. The databases that you import must not already exist because dbimport creates the databases. Therefore, you should first *drop* the existing databases using either the INFORMIX-OnLine utility *dbaccess* or other utilities.

Figure 256 shows the use of dbimport to import the first database, inf5testdb01, in the dbspace rootdbs as default.

```
$ dbimport -i /usr inf5testdb01
{ DATABASE inf5testdb01  delimiter | }

grant dba to "informix";

{ TABLE "informix".emp row size = 47 number of columns = 8 index size = 0 }
{ unload file name = emp____100.unl number of rows = 62936 }

create table "informix".emp
  (
    empno integer,
    ename char(10),
    job char(9),
    mgr integer,
    hiredate date,
    sal decimal(10,2),
    comm decimal(9,0),
    deptno integer
  );

revoke all on "informix".emp from "public";

grant select on "informix".emp to "public" as "informix";
grant update on "informix".emp to "public" as "informix";
grant insert on "informix".emp to "public" as "informix";
grant delete on "informix".emp to "public" as "informix";
grant index on "informix".emp to "public" as "informix";

dbimport completed
$
```

Figure 256. Use INFORMIX-OnLine dbimport to Complete Database Restore

Repeat the above process for the second database, inf5testdb02.

11.3.9 Partial Online Backup (Raw) Using tbunload and ADSM

This example shows the use of tbunload and ADSM to back up two tables (no logs) and then ADSM and tbload to restore the two tables. Tbunload creates a binary file from the contents of each table. The unload can be done at the table or database level. The database server can be online because tbunload locks the tables so that a transaction cannot update them during the unload.

Figure 257 on page 402 shows the use of tbunload to unload two tables, emp and emp2, from two databases, inf5testdb01 and inf5testdb02. The emp table belongs to the inf5testdb01 database, and the emp2 table belongs to the inf5testdb02 database. Please note that before you issue the tbunload command, you must create the destination files you will specify. The destination files can be created with the AIX *touch* command. Also note that for each table you want to unload, a separate tbunload command must be issued.

```
$ touch /home/informix5/emp.tbunload
$ tbunload -t /home/informix5/emp.tbunload inf5testdb01:emp
Please mount tape and press Return to continue ...
Please label this as tape number 1 in the tape sequence.
$
$ touch /home/informix5/emp2.tbunload
$ tbunload -t /home/informix5/emp2.tbunload inf5testdb02:emp2
Please mount tape and press Return to continue ...
Please label this as tape number 1 in the tape sequence.
$
$ ls -al /home/informix5
total 15824
drwxr-xr-x   2 informix informix     512 May 17 11:12 .
drwxr-xr-x  11 bin      bin          512 May  4 11:41 ..
-rw-r--r--   1 informix informix 3571712 May 17 11:12 emp.tbunload
-rw-r--r--   1 informix informix 4521984 May 17 11:14 emp2.tbunload
$
```

Figure 257. Create INFORMIX-OnLine tbunload Destination Files

Now we use the ADSM selective command to back up the files that the tblunload utility has created. We specify that each file created by tbunload gets the extension, .tbunload. The format of the command is:

```
dsmc selective -passowrd=mars -quiet /home/informix5/*.tbunload
```

To restore the tables, we first use ADSM to restore the files and then the tbload utility to load the tables to the databases. The ADSM commands we used to restore the files are:

```
dsmc restore -passowrd=mars -quiet /home/informix5/emp.tbunload
dsmc restore -passowrd=mars -quiet /home/informix5/emp2.tbunload
```

Before using tbload, we must create the databases we want to restore, either using the INFORMIX-OnLine dbaccess utility or other SQL commands. This is different from how the dbimport utility worked, where the database you want to restore must not exist.

Figure 258 on page 403 shows how we used tbload to load our two tables, emp and emp2, to our two databases, inf5testdb01 and inf5testdb02.

```
$ tbload -t /home/informix5/emp.tbunload inf5testdb01:emp
Please mount tape and press Return to continue ...
The load has successfully completed.
$ tbload -t /home/informix5/emp2.tbunload inf5testdb02:emp2
Please mount tape and press Return to continue ...
The load has successfully completed.
$
```

Figure 258. Use INFORMIX-OnLine tbload to Load Tables Back to the Databases

11.4 Informix-Online Extended Parallel Server (XPS)

The Informix-Online Extended Parallel Server Version 8.0 (XPS) is a parallel processing version of INFORMIX-OnLine. It exploits the emerging technology of loosely coupled or shared-nothing computing architectures (as shown in Figure 259 on page 404) including clusters of symmetric multiprocessors (SMPs), for example, the IBM RISC System/6000 Model J30, and massively parallel processors (MPPs), such as the IBM RISC System/6000 Scalable POWERparallel Systems (SP2).

XPS is designed to support tasks which require large amounts of data such as data warehousing and decision support. It provides a scalable approach by spreading the processing load across multiple processors or systems and therefore making the best use of available resources.

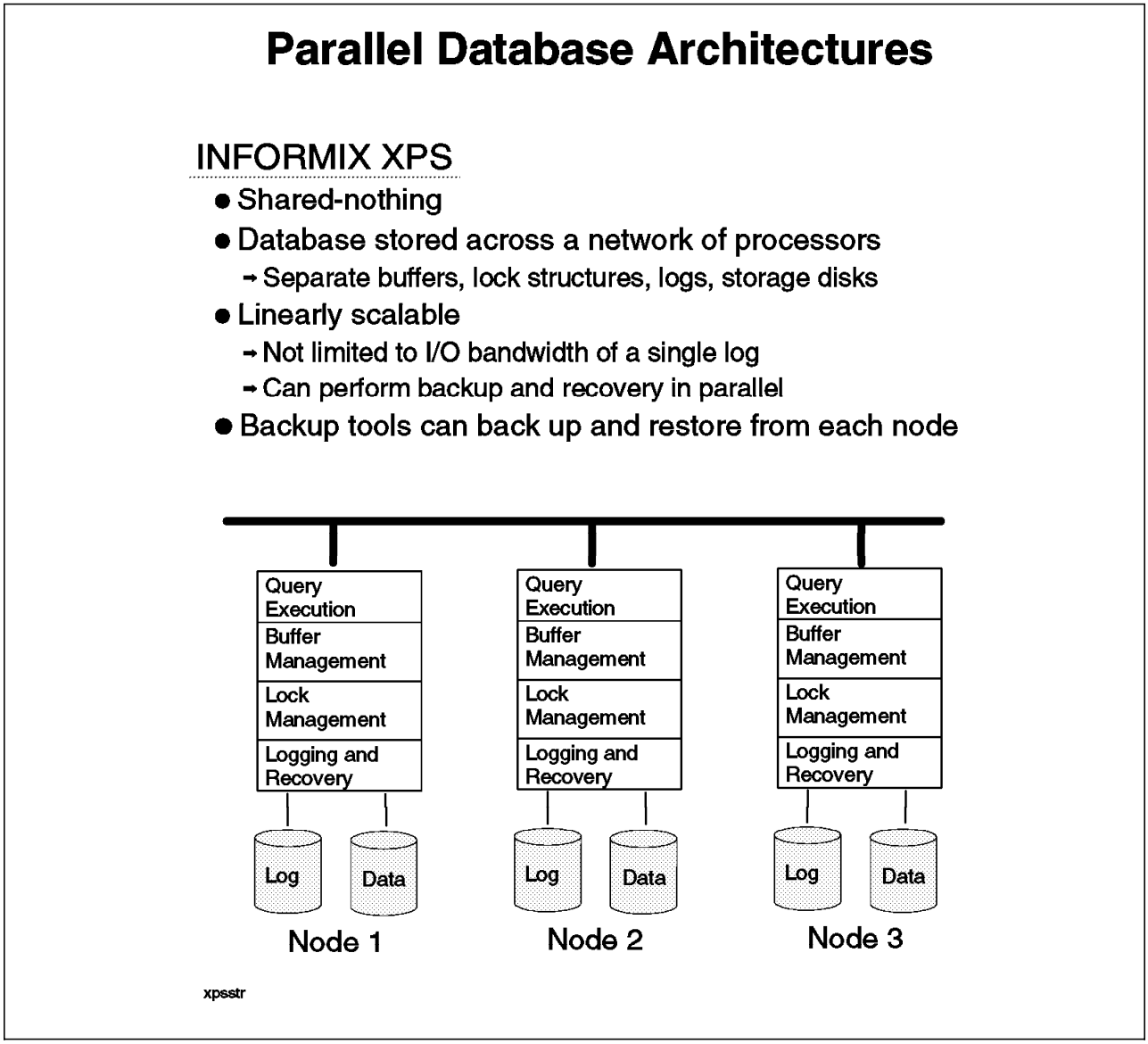


Figure 259. Shared-Nothing Parallel Database Architecture

Special thought is required when backing up data in such an environment. Data contained within one database can now be stored on disks that are “owned” by different “nodes,” and backup tools to back up the data within the database must be heavily integrated with the database manager itself.

The tool designed to solve this problem for XPS is ON-Bar, which provides an interface between the Online XPS system and a storage manager such as ADSM. As described in 11.2.4, “ON-Bar” on page 332, ON-Bar uses the XBSA and can therefore work with any storage manager that uses this standard.

Although XPS backup and restore differ from non-XPS backup and restore, INFORMIX has tried to hide the differences from the end user. Please see the *INFORMIX-OnLine XPS Backup and Restore Guide* for details.

Unfortunately, the XPS software was not available at the time of writing, so we did not test any scenarios with the ON-Bar utility and XPS.

Chapter 12. ADSM and Sybase

This chapter describes in detail how to back up Sybase databases with ADSM. First we look at Sybase′s DBMS structure so that you can understand which files exist and which files you have to factor in to your backup strategy. Then we look at the Sybase native backup utilities, providing specific examples of how to use ADSM to back up Sybase. We include a full, online database backup with Sybase′s dump database utility and ADSM and a full, offline database backup using ADSM directly. Information on the BMC Software Inc. SQL-BackTrack product which can be used to back up Sybase is described in Chapter 9, “SQL-BackTrack” on page 229. Please note that when we refer to Sybase in this book we are referring to the Sybase SQL Server.

The procedures in this chapter have been tested with the latest version of Sybase (Version 10.02) under AIX V4.1.3.

12.1 Sybase DBMS Structure

Figure 260 shows the key components of a Sybase database. Those components which you must consider for backup and recovery are:

- The master database
- The user databases
- Other databases, such as model, system procedure, and audit
- Transaction logs.

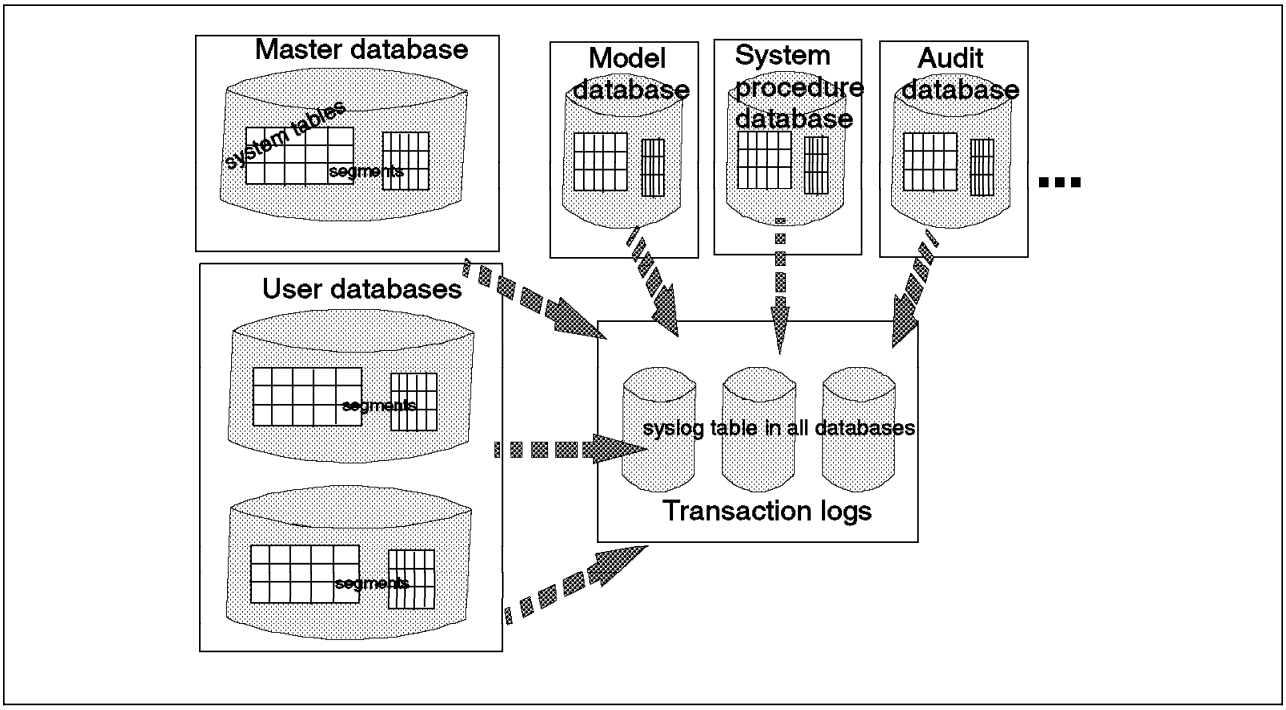


Figure 260. Sybase DBMS Structure

The **master database** controls the user databases and the operation of the server as a whole. It contains **system tables** that keep track of server information, such

as users, databases, storage space, and locks. System tables can also be thought of as the data dictionary or system catalog.

The **user databases** contain all of the user data as well as a subset of the system tables with information specific to that database.

The **model database** is used as a template for creating user databases. The **system procedure database**, called sybsystemprocs, holds system procedures. The **audit database**, called sybsecurity, is used for security.

Each database contains **transaction logs**, which are really system tables called *syslogs*. Syslogs automatically record every transaction issued by each user of the database. By default, the transaction logs for a database are stored on the same device as the database. Run the **sp_helpdb** script to find out if your logs are on the same device as your database. If they are, you do not have to back them up separately.

Segments, another Sybase component, are subsets of the database "devices" available to a particular server database. Each database can have up to 32 segments. Segments provide a flexible tool for assigning objects to particular database devices.

12.2 Sybase Backup Utilities

Sybase provides **dump database** and **dump transaction** backup utilities and corresponding **load database** and **load transaction** restore utilities. It also has a utility, **dbcc**, which allows you to check the consistency of a database or table. It is recommended that you run dbcc against a database *before* you take a backup to ensure that you are backing up a consistent database.

Dump database takes database images of the entire database, including both the data and transaction logs. It allows dynamic dumps; users can continue to use their applications to query or update database data while the dump takes place. In other words, it is an online dump utility.

Dump transaction copies the transaction log, providing a record of any database changes made since the last database or transaction log dump. It works like an incremental backup by copying all transactions which have been committed since the last dump transaction. So, for example, you might take a full database backup once a week using dump database, but then only back up the transaction logs during the week with dump transaction. Dump transaction, like dump database, works in online mode.

The dump transaction option *with no_truncate* is useful because it dumps transaction logs up to the time of a database failure if the transaction logs and database are on separate devices.

Note: If you want to use the full dump transaction command capability, you must move the transaction log to a different device from the rest of the database by using the *log on* clause of the *create database* SQL Server utility.

Load database loads the dump into an existing or new database, and load transaction rebuilds the database by reexecuting changes recorded in the transaction log. The load commands are executed while the database is offline.

In 12.3, “Using ADSM to Back Up Sybase” on page 408, we show how to use dump database and dump transaction to create file system files and then use ADSM to back up those file system files. Dump database and dump transaction do not work with the ADSM API. The future direction is for the planned Sybase System 11 Backup Server to work with the ADSM API.

Sybase also provides a **bulk copy utility**, which is not designed as a backup utility but can be used to supplement your backup strategy. The bulk copy utility is used to transfer data between the server and the operating system. It is frequently used to copy data into another program, such as another DBMS or spreadsheet. The bulk copy utility is similar to the export and import utilities provided by other RDBMSs.

The bulk copy utility is the only Sybase mechanism to copy data at a table level; the dump utilities work at the database level. You could bulk copy tables to file system files and then use ADSM to back up the file system files. We do not show an example of using the bulk copy utility. Keep in mind that you must be very careful when using an alternative like this because you run the risk of introducing database inconsistencies.

IBM and BMC Software Inc. have announced the integration of the **SQL-BackTrack** product with ADSM. Details about the SQL-Backtrack product can be found in Chapter 9, “SQL-BackTrack” on page 229.

SQL-BackTrack provides additional flexibility over the native Sybase dump database backup utility. For example, with SQL-BackTrack you can do incremental and object-level backups, object-level recovery, as well as hardware and operating system independent backups.

To obtain this support, customers purchase SQL-BackTrack and the SQL-BackTrack ADSM Module.

A detailed matrix of the Sybase backup alternatives is shown in Figure 381 on page 567.

No matter which utility or product you choose to use, here are some suggestions for your backup strategy:

- Take backups of the master database frequently. Remember that the master database contains all vital server information, so you should back it up each time you make changes to your environment. For example, each time you create, alter, or drop any device, database, or database object, and each time you execute a stored procedure that changes the database, you should back it up. The master database uses a special recovery utility, **buildmaster** which, when used, is called a **full recovery** operation.
- Take backups of the following system tables using the bulk copy utility:
  - *Sysusages*, which contains one row for each disk allocation piece assigned to a database
  - *Sysdatabase*, which contains one row for each database on the server
  - *Sysdevices*, which contains one row for each tape dump device, disk dump device, disk for the databases, and disk partition for the databases
  - *Sysloginroles*, which contains a row for each instance of a server login processing a system-defined role
  - *Syslogins*, which contains a row for each valid server user account.

If you have a copy of these tables and a crash occurs, you can rebuild your server without performing a full recovery.

12.3 Using ADSM to Back Up Sybase

In this section we examine two ways in which you can use ADSM to back up Sybase databases. We cover an online database backup using dump database and ADSM and an offline backup using ADSM directly.

Remember that if you use a Sybase utility before using ADSM, you can back up databases installed on either file system files or raw devices. If you use ADSM to back up the database directly, the database must be installed on file system files. If the database is installed on raw devices, other alternatives are to first use the AIX dd command and then use ADSM to back up the resultant file (as described in 4.5, “Backing up Databases using dd and tar” on page 64), or use the ADSMPIPE utility, which sends the data directly to ADSM through the ADSM API (as described in Appendix C, “ADSMPIPE: Raw Logical Volume Backups with the API” on page 553). ADSMPIPE and dd are offline, full backup alternatives. For more details on these technique, see 4.2, “Techniques for Using ADSM to Back Up Databases” on page 51.

12.3.1 Online Database Backup Using Dump Database and ADSM

This example shows how to use the Sybase dump database utility and then ADSM to back up the resultant files. This alternative can be used for databases installed on either file system files or raw devices, but our example shows file system files. Remember that dump database is dynamic (online) and backs up the entire database and transaction log.

12.3.1.1 Preparatory Steps

Before you back up a database using the dump database utility, you must be sure that the database has integrity. Use the **dbcc** command to check database consistency. The *dbcc* command checks the logical and physical consistency of a database. It should be used periodically or when database damage is suspected.

Use this command to check our sample database, dbtest, for consistency:

```
1> dbcc checkdb (dbtest)
2> go
```

12.3.1.2 Back up Database and Transaction Log with Sybase and ADSM

Here are the steps to back up the database and transaction log using dump database and ADSM commands:

1. Log in to AIX as operator or higher Sybase authority. .
2. Start the *isql* interface.

   ```
   isql -U sa -P  -S sybase
   ```

3. Back up your user database.

   ```
   1> dump database dbtest to "/home/sybase/dbtest_dump"
   2> go
   ```

Figure 261 on page 409 shows the results of entering these commands.

```
Backup Server session id is:  42.  Use this value when executing the
'sp_volchanged' system stored procedure after fulfilling any volume
change request from the Backup Server.
Backup Server: 4.41.1.1: Creating new disk file /home/sybase/dbtest_dump
Backup Server: 6.28.1.1: Dumpfile name 'dbtest94167OD1BD ' section
number mounted on disk file '/home/sybase/dbtest_dump'
Backup Server: 4.58.1.1: Database dbtest: 264 kilobytes DUMPed.
Backup Server: 3.43.1.1: Dump phase number 1 completed.
Backup Server: 3.43.1.1: Dump phase number 2 completed.
Backup Server: 3.43.1.1: Dump phase number 3 completed.
Backup Server: 4.58.1.1: Database dbtest: 272 kilobytes DUMPed.
Backup Server: 3.42.1.1: DUMP is complete (database dbtest).
```

Figure 261. Results of Sybase Dump Database Command to Back Up Server

4. Back up the transaction log.

   The dump transaction command makes a copy of the transactions from the last dump transaction or dump database command for the selected database.

   Note: If you have a database failure and you have lost your database device then you should perform a dump transaction command with the *with no_truncate* **before** you start to recover your database. This will back up the current transaction log which will be needed when you recover your database. The basic dump transaction command is:

   ```
   1> dump transaction dbtest to "/home/sybase/dbtest_tr01"
   2> go
   ```

   Figure 262 shows the results of entering these commands.

```
Backup Server session id is:  46.  Use this value when executing the
'sp_volchanged' system stored procedure after fulfilling any volume
change request from the Backup Server.
Backup Server: 4.41.1.1: Creating new disk file /home/sybase/dbtest_tr01
Backup Server: 6.28.1.1: Dumpfile name 'dbtest94167OD346 ' section
number mounted on disk file '/home/sybase/dbtest_tr01'
Backup Server: 4.58.1.1: Database dbtest: 8 kilobytes DUMPed.
Backup Server: 3.43.1.1: Dump phase number 3 completed.
Backup Server: 4.58.1.1: Database dbtest: 12 kilobytes DUMPed.
Backup Server: 3.42.1.1: DUMP is complete (database dbtest).
```

Figure 262. Results of Sybase Dump Transaction Command to Back Up Log

5. Use ADSM to back up your database image files.

   Selected files are backed up to the ADSM server. With the dump database and dump transaction commands, we chose to call the output files related to our *dbtest* database using a common sequence of characters (dbtest). Figure 263 on page 410 shows the ADSM command to back up the files and the results of that command.

```
dsmc selective /home/sybase/dbtest* -password=mars

Session established with server BALTIC: AIX-RS/6000
 Server Version 1, Release 2, Level 0.0
 Server date/time: 06/16/1994 15:02:53   Last access: 06/16/1994 10:11:0

Normal File-->         288,768 /home/sybase/dbtest_dump .. Sent
Selective Backup processing of '/home/sybase/dbtest_dump' finished with no failures

Normal File-->          22,528 /home/sybase/dbtest_tr01 .. Sent
Selective Backup processing of '/home/sybase/dbtest_tr01' finished with no failures
```

Figure 263. ADSM Backup of Sybase Dump Command Output

12.3.1.3 Restore Database and Transaction Log Using ADSM and Sybase

If you want to restore a previously saved dumped image because of a system crash or database failure, use ADSM to restore the image files onto the workstation and then use Sybase load to re-create it.

1. Figure 264 shows the ADSM command to restore database image files and the results of that command.

```
dsmc restore "/home/sybase/dbtest*" -password=mars

Restore function invoked.

Session established with server BALTIC: AIX-RS/6000
 Server Version 1, Release 2, Level 0.0
 Server date/time: 06/16/1994 15:20:55   Last access: 06/16/1994 15:08:3

Restoring      288,768 /home/sybase/dbtest_dump ... Done
Restoring       22,528 /home/sybase/dbtest_tr01 . Done

Restore processing finished.
```

Figure 264. ADSM Restore of Sybase Dump Command Output

2. Start the isql interface.

   ```
   isql -Usa -P  -Ssybase
   ```

3. Restore the database using Sybase load.

   Before starting the load utility, make sure that no users are connected to the database you want to restore by using *sp_dboption* to set the *dbo use only* and *read only* options to **TRUE**. Figure 265 on page 411 shows this.

```
1> use master
2> go
1> sp_dboption dbtest,"read",true
2> go
Database option 'read only' turned ON for database 'dbtest'.
(return status = 0)
1> sp_dboption dbtest,"dbo use only",true
2>go
Database option 'dbo use only' turned ON for database 'dbtest'
(return status = 0)
```

Figure 265. Set Sybase Database to Read Only

Now run the dump database command to restore the database dump:

```
1> load database dbtest from "/home/sybase/dbtest_dump"
2> go
```

Figure 266 shows the results of the command.

```
Backup Server session id is:  49.  Use this value when executing the
'sp_volchanged' system stored procedure after fulfilling any volume
change request from the Backup Server.
Backup Server: 6.28.1.1: Dumpfile name 'dbtest94167OD1BD ' section
number mounted on disk file '/home/sybase/dbtest_dump'
Backup Server: 4.58.1.1: Database dbtest: 6148 kilobytes LOADed.
Backup Server: 4.58.1.1: Database dbtest: 6156 kilobytes LOADed.
OO:94/06/16 15:24:5O.12 server  Recovery dbid 8 ckpt (2O57,27)
OO:94/06/16 15:24:5O.18 server  Recovery no active transactions before
Backup Server: 3.42.1.1: LOAD is complete (database dbtest).
```

Figure 266. Restore Sybase Database Using Load Command

4. Restore the transaction log.

```
1> load transaction dbtest from "/home/sybase/dbtest_trO1"
2> go
```

Figure 267 shows the results of the load transaction command.

```
Backup Server session id is:  52.  Use this value when executing the
'sp_volchanged' system stored procedure after fulfilling any volume
change request from the Backup Server.
Backup Server: 6.28.1.1: Dumpfile name 'dbtest94167OD346 ' section
number mounted on disk file '/home/sybase/dbtest_trO1'
Backup Server: 4.58.1.1: Database dbtest: 12 kilobytes LOADed.
Backup Server: 3.42.1.1: LOAD is complete (database dbtest).
```

Figure 267. Restore Sybase Log Using Load Command

12.3.2 Offline Database Backup Using ADSM Directly

This example shows how to use ADSM to back up the Sybase database and log files directly. This alternative can be used only for databases installed on file system files. The database must be offline to ensure database consistency. The advantage of this approach is that no additional disk space is needed to store the results from the Sybase dump utilities. Remember to use the ADSM MAKESParsefile=no parameter during restoration to properly handle sparse files. See 4.3.1, “Handling Sparse Files” on page 56 for more details.

12.3.2.1 Preparatory Steps

Before you can use ADSM to back up Sybase databases directly, you must check the integrity of the database with the dbcc command.

If you want to use ADSM commands to back up a database directly, make sure no one is using that database. Use the *sp_who* Sybase command to verify this:

```
1> sp_who
2> go
```

Figure 268 shows the results of the command.

```
spid  status       loginame     hostname   blk   dbname     cm
----- ------------ ------------ ---------- ----- ---------- ---------
    1 running      sa           baltic     0     master     SELECT
    2 sleeping     NULL                    0     master     NETWORK HAND LER
    3 sleeping     NULL                    0     master     MIRROR HANDL ER
    4 sleeping     NULL                    0     master     CHECKPOINT SLEEP
```

Figure 268. Sample Output from Sybase sp_who Command

To determine which files to back up with ADSM, you must first list all of the “devices” that the database uses and then issue an SQL query to one of the master system tables.

Use *sp_helpdb* to list the database devices:

```
1> sp_helpdb dbtest
2> go
```

Figure 269 shows the results of the command.

```
device_fragments      size          usage                free kbytes
--------------------- ------------- -------------------- -----------
dev1                  2.0 MB        data only                   1312
dev2                  2.0 MB        log only                    2032
log_dev               2.0 MB        data only                   2048
```

Figure 269. Sample Output from Sybase sp_helpdb Command

The *device_fragments* column shows the logical name that Sybase uses to manage storage associated with a database. To find the physical name where data is written to, you must look in the *sysdevices* table of the *master* database. To do this, run the following SQL statement:

```
1> select name,phyname from master..sysdevices
2> go
```

The result, shown in Figure 270 on page 413, is a list of devices defined and known by the SQL server. Write down the physical devices that belong to your database. For our example, the highlighted devices, dev1, dev2, and log_dev, are the devices belonging to our dbtest database.

```
dev1       /usr/sybase4.10/p_dev1.d
dev2       /usr/sybase4.10/p_dev2.dbs
log_dev    /usr/sybase4.10/p_log_te
master     d_master
raw_dis    /dev/rdb00
tapedump   /dev/rmt
tapedump2  /dev/rst0
```

Figure 270. Database Files Belonging to Sybase Devices

12.3.2.2 Using ADSM to Back Up Database Files

1. Shut down the Sybase SQL server.

   ```
   1> shutdown
   2> go
   ```

   You must shut down the server because the database you want to save is active in the SQL server even if no users are connected to it. Part of the database may be loaded in internal buffers, and the ADSM command would not save the data properly if the database was not shut down.

2. Use ADSM dsmc command to back up the database files.

   Use the *dsmc* command to back up database files to the ADSM server. Note that all files that belong to the *testdb* database start with *p_*. We chose this naming convention when creating the database so that we could immediately identify all of the database components if there were many databases.

   Figure 271 shows the ADSM dsmc command to back up the database and the results of that command.

```
dsmc selective "/usr/sybase4.10/p_*" -password=mars

Selective Backup function invoked.

Session established with server BALTIC: AIX-RS/6000
 Server Version 1, Release 2, Level 0.0
 Server date/time: 06/16/1994 16:06:02  Last access: 06/16/1994 16:05:55

Normal File-->       2,097,152 /usr/sybase4.10/p_dev1.dbs .. Sent
Normal File-->       2,097,152 /usr/sybase4.10/p_dev2.dbs . Sent
Normal File-->       2,097,152 /usr/sybase4.10/p_log_testdb . Sent
Selective Backup processing of '/usr/sybase4.10/p_*' finished with no
failures.
```

Figure 271. ADSM Command to Back Up Sybase Database Files

3. Restart the Sybase SQL server.

   ```
   sybase/install/startserver -f $sybase/install RUN_sybase
   ```

12.3.2.3 Using ADSM to Restore Database Files

To restore a database previously backed up with ADSM commands directly, you have to shut down the SQL server and use ADSM commands to restore the database files.

1. Shut down the Sybase SQL server.

```
1> shutdown
2> go
```

2. Use ADSM to restore the database files. Figure 272 shows the command and the results from it.

```
dsmc restore "/usr/sybase4.10/p_*" -password=mars

Restore function invoked.

Session established with server BALTIC: AIX-RS/6000
 Server Version 1, Release 2, Level 0.0
 Server date/time: 06/16/1994 16:13:01   Last access: 06/16/1994 16:11:1

File /usr/sybase4.10/p_dev1.dbs already exists, do you want to replace it (Yes/No)
Restoring    2,097,152 /usr/sybase4.10/p_dev1.dbs ... Done

File /usr/sybase4.10/p_dev2.dbs already exists, do you want to replace it (Yes/No) y
Restoring    2,097,152 /usr/sybase4.10/p_dev2.dbs . Done

File /usr/sybase4.10/p_log_testdb already exists, do you want to replace it (Yes/No) y
Restoring    2,097,152 /usr/sybase4.10/p_log_testdb . Done

Restore processing finished.
```

Figure 272. ADSM Command to Restore Sybase Database Files

3. Restart the Sybase database.

```
sybase/install/startserver -f $sybase/install RUN_sybase
```

Part 4. DB2 Backup and Recovery

Chapter 13. ADSM and DB2/6000

This chapter presents an overview of the major components of DB2/6000 and describes how you can use ADSM to back up DB2/6000 databases. First we look at the DB2/6000 DBMS structure and identify which files and devices have to be considered for backups. Thereafter, we look at the DB2/6000 Versions 1 and 2 backup utilities and give sample scenarios of backup operations. These scenarios include full, offline and full, online database backups. We also describe backups of table spaces using the enhanced utilities of DB2/6000 Version 2. The output of the backup utilities is sent directly to ADSM, without creating any intermediate files.

We finish with some additional considerations, such as querying the ADSM server, deleting backup copies stored in ADSM with a new DB2/6000 V2.1.1 utility (db2adutl), and DB2/6000 Parallel Edition (DB2PE).

Even though this chapter focuses on DB2/6000, all of the information herein is applicable to DB2 common server on all UNIX platforms. The path in which DB2 common server and the ADSM Client are installed is platform specific.

13.1 DB2/6000 DBMS Structure

The major components of DB2/6000 include the database engine and facilities to access data, such as the command line processor, the Database Director, and application interfaces. The database engine provides the base functions to manage your data. It controls all access to it, provides transaction management, and ensures data integrity and data protection. The basic elements of the database engine are the:

- Database
- Configuration files
- Directories
- Recovery logs.

Figure 273 on page 418 shows these key components.

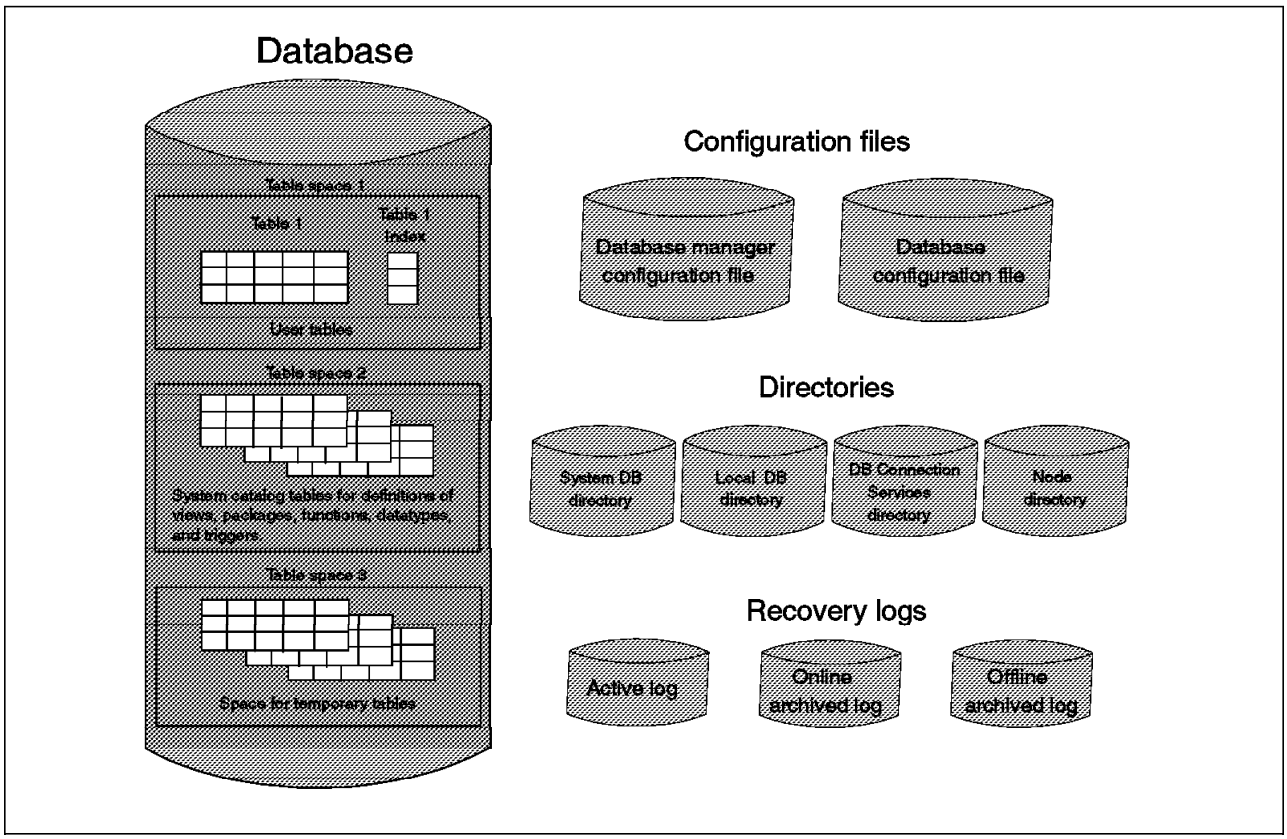


Figure 273. DB2/6000 DBMS Structure

13.1.1 Database

Each **database** includes a set of *system catalog tables* and a collection of user tables. DB2/6000 Version 2 allows you to partition databases into parts called *table spaces*. When creating a table, you can decide to have certain objects, such as indexes and large object data, kept separately from the rest of the table data. DB2/6000 creates and maintains an extensive set of system catalog tables for each database. These tables contain information about database objects such as tables, views, packages, referential integrity relationships, functions, distinct types, and triggers. The system catalog tables are created when the database is created and updated during the normal operation. In addition, the system catalog tables describe the logical and physical structure of the data. The database manager creates a separate subdirectory when a database is created. However, objects associated with the database are not always stored in the database directory. You can specify that objects be stored in various locations, including raw devices.

Note: DB2/6000 Version 1 can only store data on a JFS. DB2/6000 Version 2 can store data directly on a raw device.

13.1.2 Configuration Files

The **configuration files** contain parameter values that define the resources allocated to DB2/6000 and to individual databases. The **database manager configuration file** is created when DB2 is installed or when an instance of DB2 is created. It contains parameter values for DB2/6000 as a whole. These parameters have globally, independent of any one database stored on the

system. On UNIX-based systems this file is called db2systm and can be found in the sqllib subdirectory for the instance of the database manager. A **database configuration file** is created when a database is created. This file resides in the SQLxxxxx directory, where xxxxx is the number assigned when the database was created. The file is named SQLDBCON. Its parameters specify the amount of resources to be allocated to that database. There is one configuration file for each individual database.

13.1.3 Directories

The **directories** are necessary for accessing local and remote databases. DB2 uses three types of database directories which identify the location of databases, and a node directory, which contains network connection information for remote databases. These directories ensure that access to a database is transparent, regardless of where the database physically resides. The **system database directory** identifies the name and physical location of each database. It resides in the sqllib directory. The **local database directory** contains the name and the path name in which the database files are stored. It resides in every subdirectory that contains a database. It is used to access the databases in that subdirectory. When a database is created, an entry is added to the local database directory. The **database connection services directory** is used with another IBM product, Distributed Database Connection Services/6000 (DDCS/6000). It contains the host databases that your node can access and resides in the $HOME/sqllib directory. The **node directory** contains an entry for all remote nodes to which your node can connect.

13.1.4 Recovery Logs

All databases have **recovery logs** associated with them which log ongoing transactions. These logs contain the changes made to the database since the last backup. In DB2/6000, the **active logs** prevent a failure from leaving the database in an inconsistent state. They contain information for transactions whose changes have not yet been written to the database files. In case of a failure, the changes already made but not committed are rolled back, and all committed transactions, which may not have been physically written to disk, are redone. These actions ensure the integrity of the database. The active logs reside by default in a directory called SQLOGDIR. Log files are numbered from S0000000.LOG to S9999999.LOG.

Roll-forward recovery uses logs to allow a database to be rebuilt to a specified point in time. In addition to using the information in the active logs to rebuild a database, the roll-forward function uses archived logs to reapply previous changes. When all changes in the active log are no longer needed for normal processing, the log file is closed, and it becomes an archived log. It is said to be an **online archived log** when it is stored in the database log path directory. It is an **offline archived log** when it is not stored in the database log path directory. You use a user exit program to archive logs.

13.2 DB2/6000 Backup Utilities

DB2/6000 Version 1 provides a utility called **backup** for online and offline backup of an entire database.

The online backup is done while the database is online. Thus users or applications can connect to the database and make changes while the backup is

running. The database image file is not consistent, and therefore it is not possible to use it alone to perform a recovery operation because active units of work are present in both log files and database tables.

For offline backup, no other processes can be connected to the database at the same time. Each application must be logged off, and when the backup operation starts, no SQL operations are permitted.

The DB2/6000 Version 1 **recovery** utility rebuilds a database from a previously saved database backup image. There are two recovery methods:

- **Restore recovery**, which rebuilds a database to its state when the backup copy was made. You can also use this function to duplicate a database.
- **Roll-forward recovery**, which builds a database from a restored copy, making changes to it since the time the backup was made. These changes are found in the log files.

To use the backup utility, the database must be a local database. Also, if you decide to direct your backup to disk media (without using ADSM), only one file is created. If the total size of your database is larger than 2GB, you cannot back up your data to an AIX file (ADSM support for greater than 2GB files is planned but was not available at the time of writing).

DB2/6000 also provides an export/import utility to move data. You can use the utility to supplement your backup strategy, but it is not a replacement for the backup utility. There is a risk of introducing inconsistencies into your database because data is not synchronized with the logs. However, the DB2/6000 Version 1 backup utilities only backs up entire databases, so use of the export/import utility is the only way to capture individual tables.

The DB2/6000 backup utilities (not export/import) are integrated with ADSM services through the ADSM API. Therefore an intermediate file is not created during the backup operation before the database image on the ADSM server is stored. Both online and offline backups can be performed with ADSM.

DB2/6000 Version 2 provides significant enhancements to the backup/restore utilities and a new load utility that is an order of magnitude faster than the existing import utility. With Version 2, databases can be partitioned into parts called **table spaces**, and backup and recovery can be performed at the table space level. If a table space contains a single table, the backup or recovery is equivalent to a table level backup or recovery. Table spaces can be online or offline during the backup process. During recovery, all table spaces, other than the one being recovered, can remain online.

Another enhancement to the backup/recovery process is **parallel backup and recovery**. It is possible to perform the backup or recovery of a database or table space in parallel to or from multiple devices. This is done by establishing multiple paths to or from backup devices such as tape drives or ADSM. You can specify the actual number of sessions as a parameter of the backup or restore utility. Multiple sessions can drastically reduce the elapsed time requirements.

A new **load** utility significantly increases the speed of doing data loads and ensures recoverability of the data being loaded. It is intended for either bulk loading of new tables or appending large amounts of data to existing tables. The load utility is restartable and recoverable. If a failure occurs while loading data, users can continue the load without starting from the beginning. To ensure

recoverability, a backup copy of the loaded table is created after the load processing. The backup copy can be sent directly to ADSM through the same ADSM API that DB2 backup uses.

13.3 Creating a Sample Database

DB2/6000 Version 2 supports databases stored on either a JFS or raw device. This section describes how to install the sample database on a JFS and raw device.

It is relatively simple to install the sample database on a JFS. To install the sample database on a JFS, run the db2sampl command. This command creates a database and tables with sample data. The syntax of the db2sampl command is documented in the *DB2/6000 Command Reference*. The contents of the tables are published in the *DB2/6000 SQL Reference*.

It is a bit more complicated to install a sample database on a raw device. These are the necessary steps:

1. Create the raw device.
2. Create the database.
3. Create the table space on the raw device.
4. Create and install the tables in the desired table space.

Let us look at each step in more detail.

13.3.1 Create the Raw Device

To create the raw device use AIX **smit**:

1. Enter smit from the command line.
2. Select *System Storage Management (Physical & Logical Storage)*.
3. Select *Logical Volume Manager*.
4. Select *Logical Volumes*.
5. Select *Add a Logical Volume*.
6. Enter the volume group name. For this example we use rootvg as the volume group.
7. Enter the logical volume name and the number of logical partitions. We use db2 as the name of the logical volume. The number of logical partitions is 1, and it is 4MB.
8. Wait until the logical partition is added and exit from **smit**.
9. Change the /dev/rdb2 file owner to the database instance:

   ```
   chown db26000.sysadm /dev/rdb2
   ```

   For this example db26000 is the instance and sysadm is the group to which the instance belongs.

13.3.2 Create the Database

Create a database using the following command:

```
db2 create database rawsampl
```

where rawsampl is the name and the alias for the database. The database is created in the default directory, which is specified in the **Database Manager Configuration**.

13.3.3 Create the Tablespace on the Raw Device

Connect to the created database:

```
db2 connect to rawsampl
```

Use the create tablespace command to create a table space on the raw device. The table can be the same size as the table space:

```
db2 create tablespace managed by database using (device '/dev/rdb2' 1000)
```

13.3.4 Create and Install the Tables in the Desired Tablespace

You can use different commands to create tables, for example, import with the create option or the create table command. For this example, we use the import command to create the employee table:

```
db2 import from emp.ixf of ixf create into EMPLOYEE
```

13.4 Customizing the Environment to Use ADSM

Before you can use ADSM with the DB2/6000 backup and restore utility, you must customize your environment. This customization is slightly different for the ADSM Version 1.2 and ADSM Version 2.1 clients. We strongly recommend that you use and ADSM Version 2.1.5 or higher client. Here are the ADSM Clients included with the packaging of different releases of DB2/6000:

DB2/6000	ADSM Client
V 1.2.x	V 1.1.0
V 2.1.0 - V 2.1.1	V 1.2.7
V 2.1.2 +	No longer including ADSM Client with DB2/6000 packaging

The ADSM Client is no longer shipped with DB2/6000 as of DB2/6000 Version 2.1.2. You must obtain and install the ADSM Client separately if you want to use ADSM. DB2 common server now supports ADSM on all platforms on which both DB2 common server and the ADSM Client are available; that is, AIX for RISC System/6000, HP-UX, Solaris for SUN, SINIX, Microsoft Windows NT, and OS/2. Even though this chapter focuses on DB2/6000, all information is applicable to DB2 common server on any UNIX platform. The location of DB2 common server and the ADSM Client is platform-specific, however (see Appendix E, "ADSM API Files: Location and Environment Variables" on page 569 for details on ADSM API file locations).

When you install DB2/6000, a directory called *adsm* is also created in the *$HOME/sqllib* directory of the instance owner. For DB2/6000 V 2.1.1 or earlier, it contains an API library, include files, and a sample source program to help you

better manage the environment. However, you must compare these files with those supplied with ADSM (if any). The ADSM files are found in */usr/lpp/adsm/api/bin*. The largest files should be used because they are the latest versions. Please read the following carefully because it can be confusing when you have different ADSM API code levels and versions:

Before the database manager can use the ADSM option, *root user* and the *instance* owner of your database should perform the following setup activities:

1. Identify the latest versions of the following files and make sure you have copies of them in the */usr/lpp/adsm/bin* directory:
   - libApiDS.a - API Library
   - dscameng.txt - message file
   - dsgameng.txt - message file
   - dsmapitca - API trusted agent.
2. To prevent mixing files from different ADSM versions, we recommend that you delete the ADSM-only files from the /usr/lpp/db2*_/adsm directory if ADSM is installed as a separate product, for example, you have a /usr/lpp/adsm directory on the node. You should also ensure that there is a symbolic link from /usr/lib/libApiDS.a to /usr/lpp/adsm/api/bin/libApiDS.a.
3. The *instance owner* of your database must set the following environment variables in either your */home/instance/.profile* or in the *db2profile* file.
   - *DSMI_DIR*

     Identifies the directory path where the agent file, *dsmapitca*, and the API message file, *dscameng.txt*, are located. A korn shell example is:

     ```
     export DSMI_DIR=/usr/lpp/adsm/bin
     ```

   - *DSMI_CONFIG*

     Identifies the full directory path and file name of the ADSM user options file, *dsm.opt*. This file contains the name of the server to be used. A korn shell example is:

     ```
     export DSM_CONFIG=/usr/lpp/adsm/bin/dsm.opt
     ```

   - *DSMI_LOG*

     Identifies the directory path where the error log file, *dsierror.log*, is to be created. A korn shell example is:

     ```
     export DSMI_LOG=/usr/lpp/adsm/bin
     ```

4. Create the *dsm.sys* file.

   The *root user* of your system must create or modify the ADSM system options file, dsm.sys, which must be located in the */usr/lpp/adsm/bin* directory.

   The sample *dsm.sys.smp* in the */usr/lpp/adsm/bin* directory provides the basic information needed for DB2/6000 to work. The *passwordaccess* parameter must be set to *generate*. This parameter specifies that ADSM encrypts and stores the user password locally and generates a new password when the old one expires. Then DB2/6000 is not required to supply a password each time it initializes a session with the ADSM server.

5. Create the *dsm.opt* file.

   The *instance owner user* can create or modify *dsm.opt*. This file must be located in the directory specified by your DSMI_CONFIG environment

variable. The sample *dsm.opt* (called dsm.opt.smp) in the */usr/lpp/adsm/bin* directory provides options to control a user session.

Two database instances on the same machine can use different ADSM servers with different characteristics. When multiple ADSM servers are used, each servers characteristics must be set up in the *dsm.sys* file. The particular server which is to be used is determined by having separate *dsm.opt* files for each ADSM server.

6. Set the ADSM password.

   Each ADSM client must have a password to access a server. For that reason, the *root user* of your system must run the executable file, *dsmapipw*, installed in the */home/instance/sqllib/adsm* directory, to establish and reset the ADSM password. If you are using multiple ADSM servers from this machine, make sure that root has the above environment variables set correctly. When executed, the *dsmapipw* program prompts you for the:

   - *Node name*, which must not be entered. Press Enter for the default, which is the host name of the machine.
   - *Old password*, which is the current password for the ADSM node stored in the server.
   - *New password*, which is the new password for the node.

13.5 Using ADSM to Back Up DB2/6000 Version 1

In this section, we examine how you can use ADSM to back up DB2/6000 Version 1 databases. We cover:

- Offline backup and recovery
- Online database backup and recovery.

We do not look at using the export/import utility, even though the backup utility only performs the backup at the database level.

13.5.1 Full Offline Backup and Recovery

The DB2/6000 Version 1 backup and recovery utility can be used from either the:

- Command line interface
- Graphical user interface

  or

- Your own C, COBOL, or Fortran language program.

Our examples use the command line and graphical user interfaces.

13.5.1.1 Preparatory Steps

Before you can start a backup or recovery operation, make sure that the database manager is online. As shown in Figure 274 on page 425, use the *ps -ef | grep db2* command to verify that the database manager is online. You can see a list of DB2/6000 Version 1 processes, such as the logger process, *db2loggr*, and the deadlock detector, *db2dlock*.

```
#ps -ef | grep db2

 db2 17084 32186   0   May 26      -  0:00 db2loggr
 root 17593    1   0   May 26      -  0:00 db2wdog
 db2 18363 32186   0   May 26      -  0:00 db2dlock
 db2 29373 33154   1 08:37:23  pts/3  0:00 grep db2
 db2 29628 33154  15 08:37:23  pts/3  0:00 ps -ef
 db2 32186 17593   0   May 26      -  0:58 db2sysc
 db2 36711 13268   0 07:59:40  pts/2  0:00 -ksh
```

Figure 274. Command to Verify DB2/6000 Version 1 DBMS Availability

Also make sure that the ADSM server is online and policies are in place.

13.5.1.2 Offline Backup Using the Command Line

This example shows an offline backup using the ADSM command line interface.

Log in to AIX as the *database administrator* or higher authority and make sure that all applications are logged off from the database you want to save. Use the following DB2/6000 command to verify that all applications are logged off:

```
db2 list applications for database ibmsampl
```

Figure 275 shows the results of this command.

```
 Auth Id  Application Name    Agent Id   Application Id          DB name
 -------- ------------------- ---------- ----------------------  --------
 DB2      db2bp               31006      *LOCAL.DB2.940618201723 IBMSAMPL
```

Figure 275. Results of DB2/6000 Version 1 List Applications Command

In our example, a user is connected to our IBMSAMPL database. Use the following DB2/6000 Version 1 *force* command to force the user off:

```
db2 force application '31006'
```

Start the backup operation from the command line.

```
db2 backup database ibmsampl use adsm
```

The *use adsm* option tells the DBMS to use the ADSM API to write the output backup file instead of using common devices.

Wait until the message shown in Figure 276 appears.

```
Backup successful. The timestamp for this backup image is:
19940602094940.
```

Figure 276. DB2/6000 Version 1 Command Line Message

The backup operation created an image file of your database, IBMSAMPL, and put it in ADSM server storage.

13.5.1.3 Offline Backup Using the GUI

This example shows an offline backup using the DB2/6000 GUI

Start the backup operation from the GUI.

1. Start the graphical database administrator (DBA) utility:

   ```
   db2adm
   ```

2. Select the System Database Directory option from the main window (not shown).
3. Select the database you want to use and click on Backup as shown in Figure 277.

Figure 277. DB2/6000 Version 1 Select Database for Offline Backup

The Backup Database window appears (Figure 278 on page 427). Select the following characteristics on this window:

- ADSM as the target device
- Offline backup
- Buffer Size option default of 1024. This option tells the DBMS to reserve memory space for data before writing to the specified device. If you have big databases, increase the default value to improve performance during the backup.

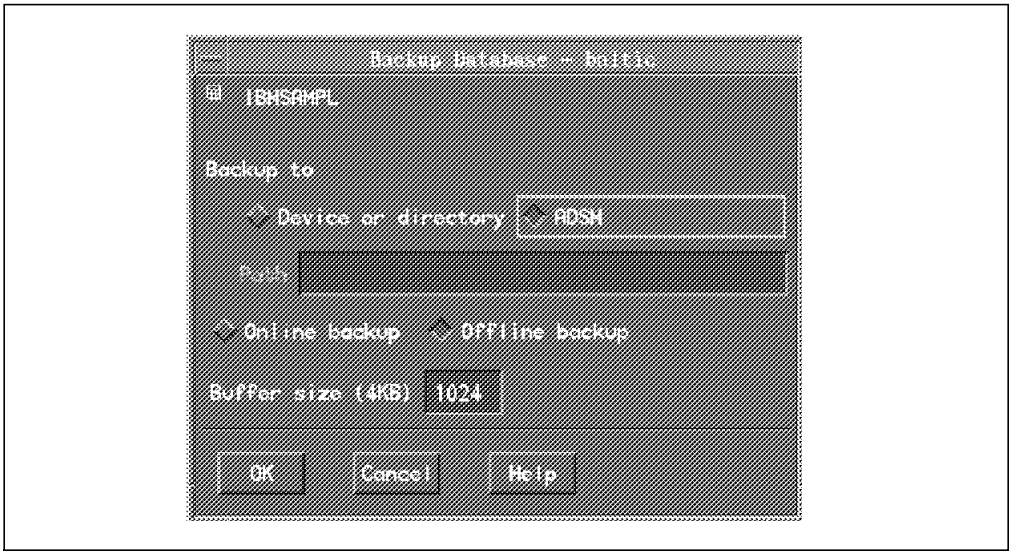


Figure 278. Select ADSM and Offline to Back Up the DB2/6000 Version 1 Database

Then click on OK to execute the backup. The Backup Progress window appears (Figure 279) and informs you of the backup state. Wait until the “The operation was successful.” message appears.

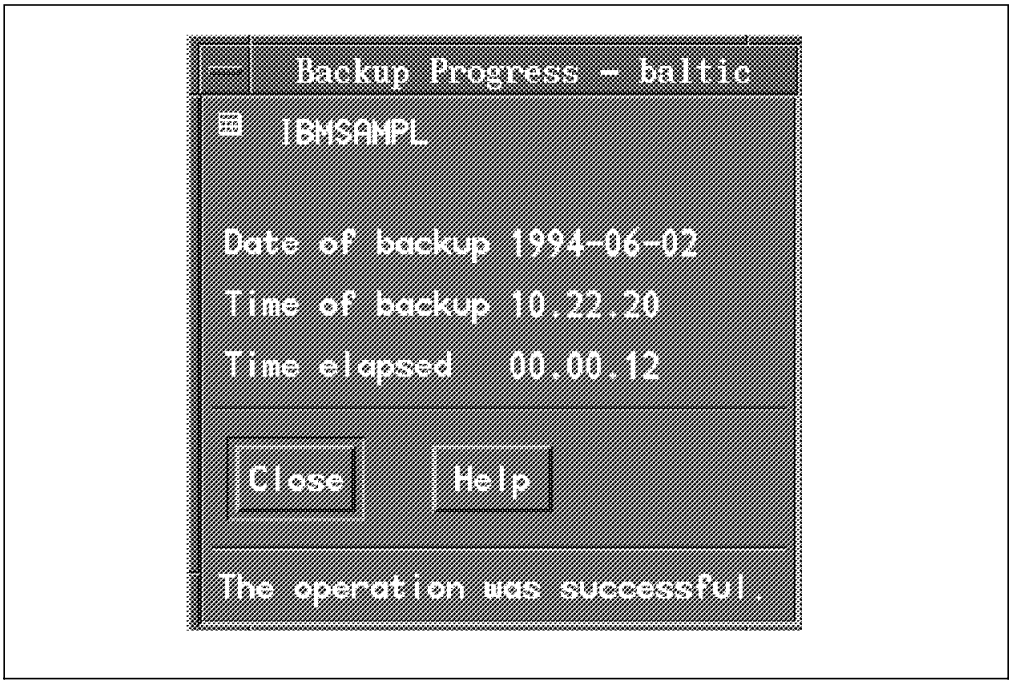


Figure 279. DB2/6000 Version 1 Offline Backup Progress Window

13.5.1.4 Recovery Example Using the ADSM Command Line

To restore a database using the ADSM command line interface follow the steps below:

Log in to AIX as the instance owner and query the ADSM server for the database backup time stamp. See 13.7.1, “Querying the ADSM Server for DB2/6000 Version 1 or DB2/6000 Parallel Edition” on page 456 for an example.

Start the recovery operation using the command line.

```
db2 restore database ibmsampl use adsm taken at 19940602102220
```

where 19940602102220 is the time stamp of the image backup file. If a time stamp is not provided then the most recent backup image is used by default.

If you are restoring the image backup file on the existing database, the restore utility asks you to confirm that choice, as shown in Figure 280. Then wait for the successful completion message to appear.

```
SQL2539W  Warning!  Restoring to an existing database that is the
same as the backup image database.  The database files will be deleted.
Do you want to continue ? (y/n) y

DB20000I  The RESTORE DATABASE command completed successfully
```

Figure 280. DB2/6000 Version 1 Command Line Restore

You can use the following additional options during the recovery process:

Target database alias, which is the alias name of the database for the recovery utility. You can specify a nonexisting database alias name, and the database will be created for you.

Buffer size, which is the size in pages of the buffer used for the restore. The default value is 1024 pages. If you have big databases, increase the default value to improve performance during the recovery.

13.5.1.5 Recovery Example Using the DB2 GUI

To restore a database using the GUI follow the steps below:

Query the ADSM server for the database backup time stamp. See 13.7.1, “Querying the ADSM Server for DB2/6000 Version 1 or DB2/6000 Parallel Edition” on page 456 for an example.

Start the graphical DBA by using:

```
db2adm
```

Select the System Database Directory from the main window (not shown). The System Database Directory appears (Figure 281 on page 429).

Select the database you want to recover and click on Recover.

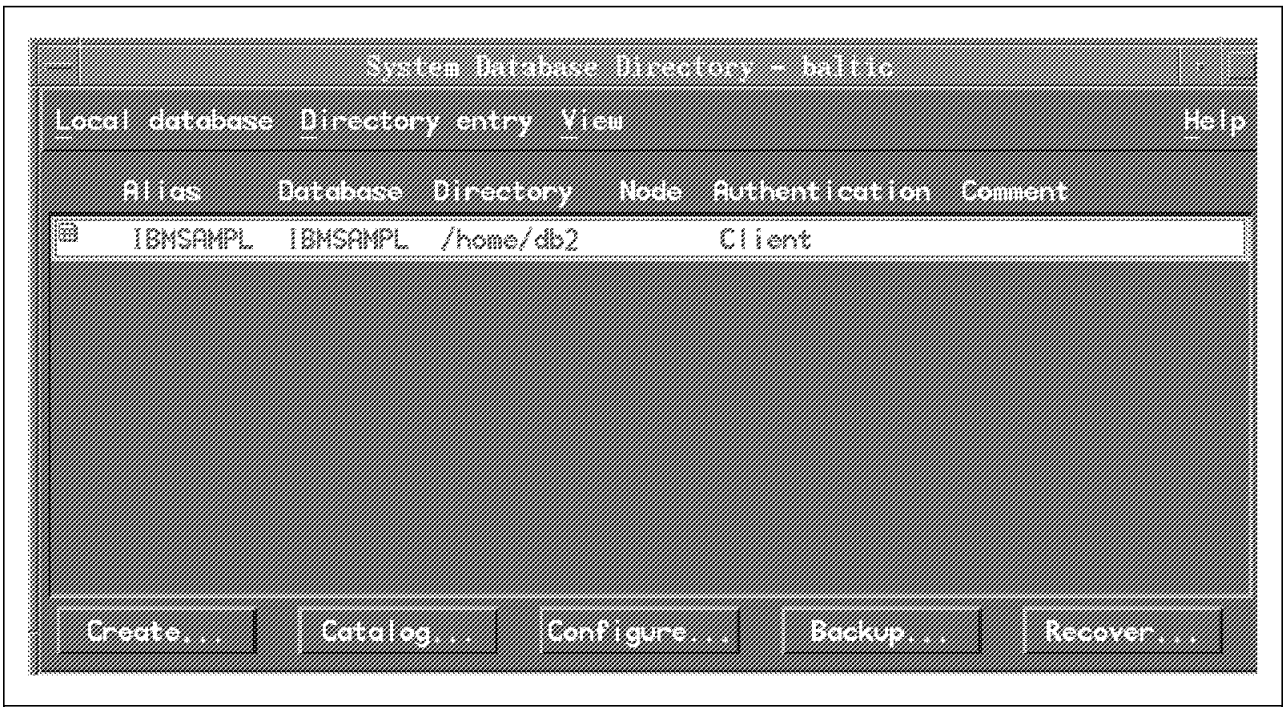


Figure 281. Select the DB2/6000 Version 1 Database to Recover

The Recover Database window appears (Figure 282 on page 430). Set the following characteristics:

- Select ADSM as the *Restore from* option.
- Specify the Alias name of the database you want to restore.
- Specify the date and time to identify which backup image to use for the restore.
- Select Existing database if you want to overwrite the existing database or New database to create a new database. If you select Existing database, the alias database name is automatically filled in. If you select new database, enter its name (next to Database) and the directory in which you want to create the new database.
- Modify the Buffer size parameter according to the database size and the *COMMTIMEOUT* parameter of the ADSM server.
- Select Roll forward if you want to perform a roll-forward recovery operation after the restore.

  In our example, we did not select the Roll forward option. We show this option in the next example in 13.5.2, “Full Online Backup and Recovery” on page 431.

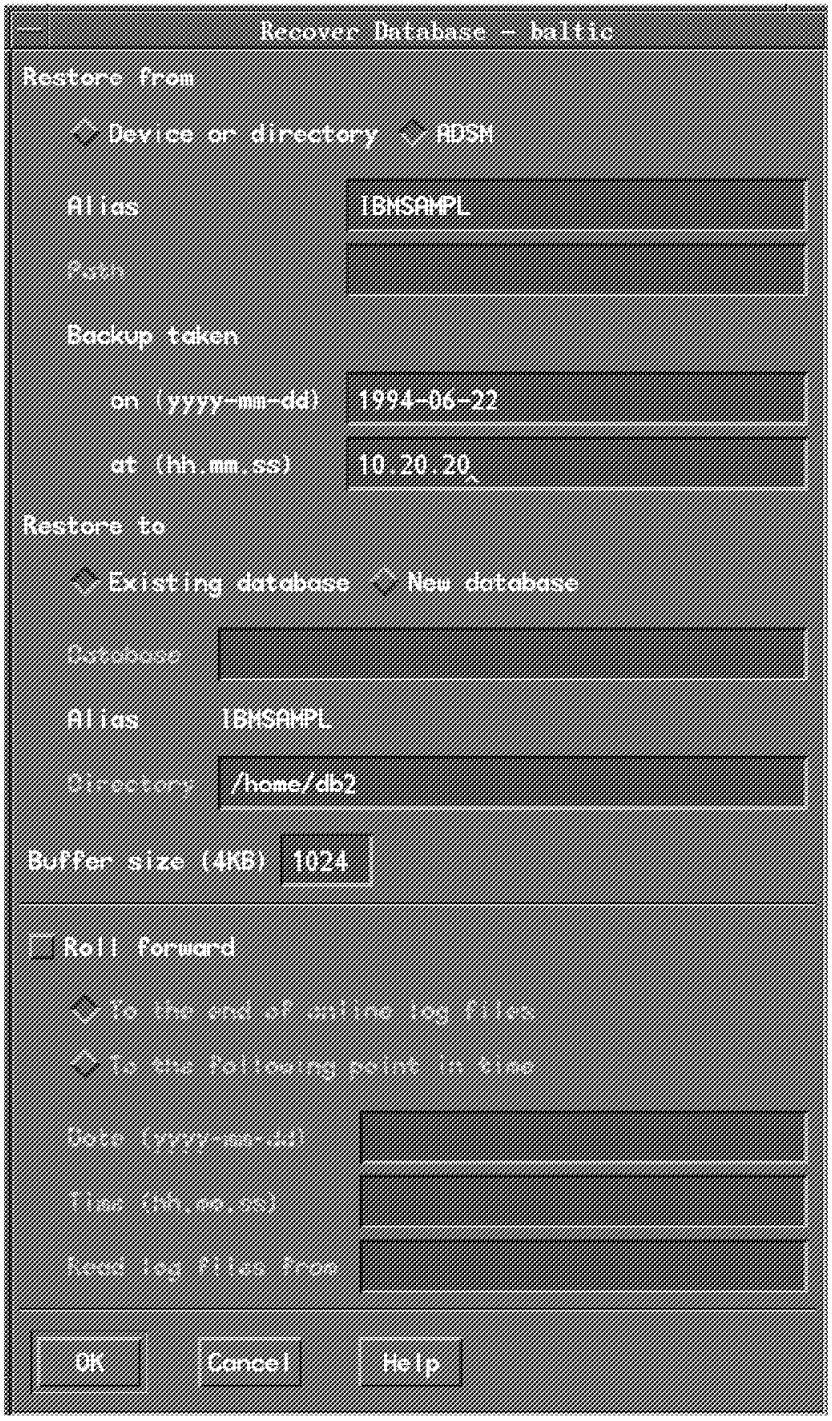


Figure 282. Select DB2/6000 Version 1 Restore Options

Click on OK to start the restore. A small window appears (not shown) informing you that a database named *IBMSAMPL* already exists and prompting you as to whether you want to proceed with the recovery operation.

The Restore Progress window appears (Figure 283 on page 431). The restore is complete when the “The operation was successful.” message appears.

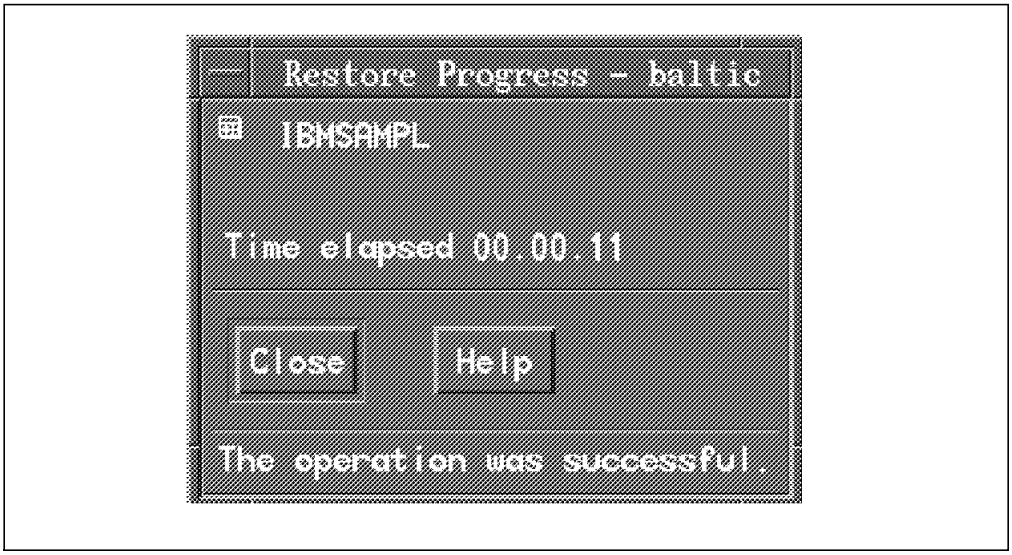


Figure 283. Restore of DB2/6000 Version 1 Database after Offline Backup

13.5.2 Full Online Backup and Recovery

DB2/6000 Version 1 online backup and recovery require the use of log files to enable roll-forward recovery. Our example shows both command line and GUI interfaces.

13.5.2.1 Preparatory Steps

Before you can use the DB2/6000 Version 1 online backup and recovery utility, you must customize your environment. For the DB2/6000 Version 1 database manager to use the ADSM option, both for backup/recovery and log management, the following set up activities must be performed:

1. Complete the preparatory steps in 13.4, “Customizing the Environment to Use ADSM” on page 422.
2. Activate the user exit program.

   The user exit is an executable file that DBMS calls for archive and retrieval of log files. It enables the DBMS to interact with storage devices that are not directly supported by the operating system (such as ADSM).

   When you install DB2/6000 Version 1, sample user exit programs are installed in the *$HOME/sqllib/samples/c* directory of the instance owner.

   Use the *db2uexit.cadsm* sample to archive and retrieve log files to ADSM. Perform the following steps:

   a. Log in to AIX as the instance owner.

   b. Copy the source sample file, *db2uexit.cadsm*, into your home directory by using the following command:

      ```
      cp sqllib/samples/c/db2uexit.cadsm db2uexit.c
      ```

   c. Set the *Installation Defined Variables* in the source C program, *db2uexit.c*, to suit your environment.

   d. Compile and link *db2uexit.c* with the following command:

      ```
      cc -o db2uexit db2uexit.c -L $HOME/sqllib/adsm \
       -l ApiDS -I $HOME/sqllib/adsm
      ```

e. As root user, copy the db2uexit file into the $HOME/sqllib/bin directory of the instance owner.

3. Customize the DB2/6000 Version 1 environment.

   a. Log in to AIX as the *instance owner*.

   b. Start the graphical DBA by using:

      ```
      db2adm
      ```

   c. Select System Database Directory from the main window and the System Database Directory appears. Highlight the database you want to use and click on Configure (not shown).

   d. The Configure Database window appears (Figure 284 on page 433). This window contains a number of database parameters, many of which we do not need to use for the backup. Therefore, use the scroll bar until you reach the log section. and set the following parameters:

      - Set log_retain_status to Enable. The log files are not reusable for writing transactions, and they are saved in the same directory in which they are created.
      - Set user_exit_status to Enable. The log files are moved as soon as they become inactive to other devices specified in the user exit program.

      Click on OK to confirm these changes. A message appears (not shown) to inform you that all applications must disconnect from the database before changes are effective.

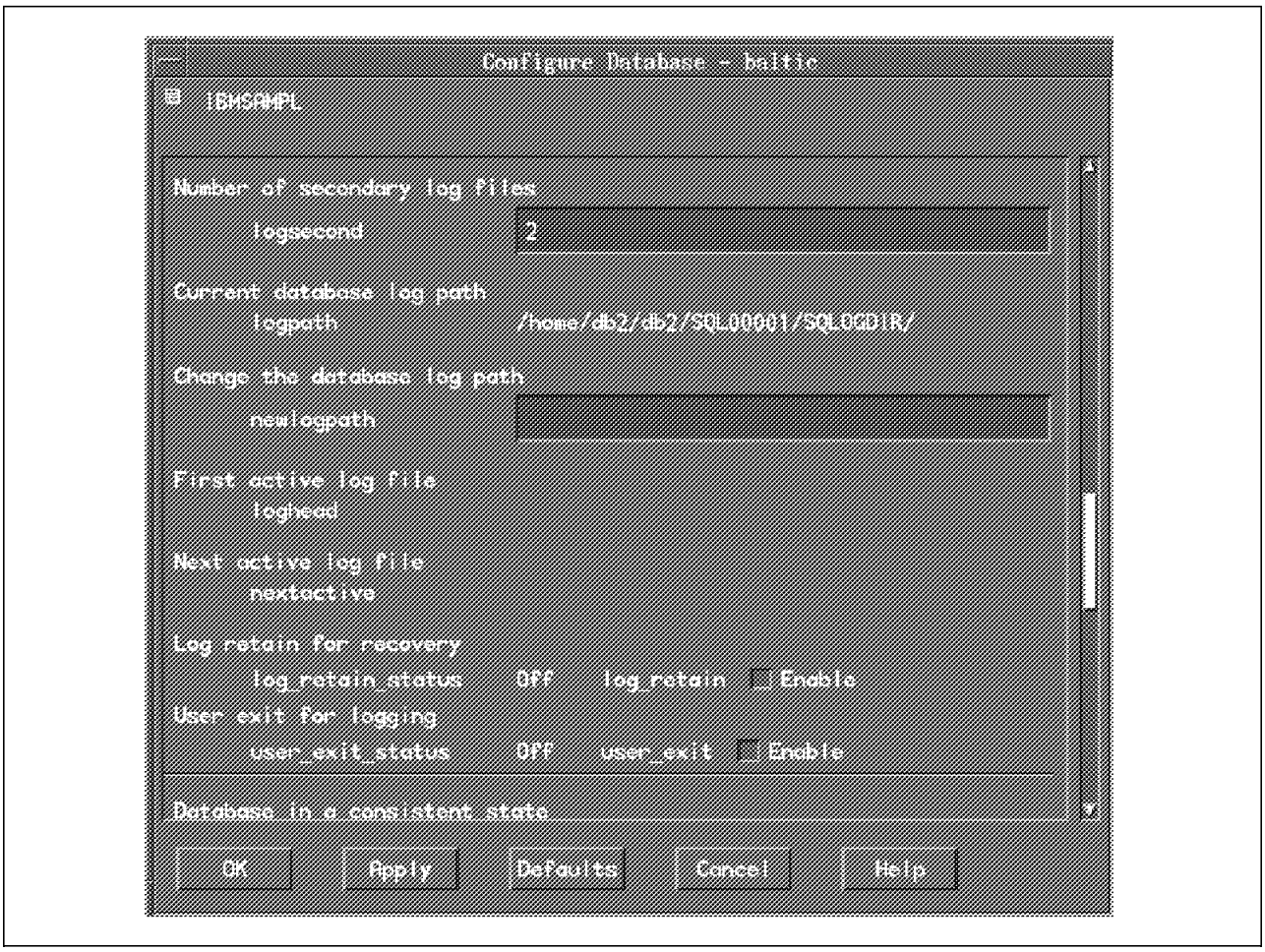


Figure 284. Configure DB2/6000 Version 1 Log File Parameters

Now you must take an offline backup before using the database to ensure that you are using the modified environment and starting from a consistent point (the offline backup). A message window appears (not shown) to inform you to take this offline backup. Click on OK to remove it.

As shown in Figure 285 on page 434, you can tell that the database is in a *backup pending state* by looking at the icons near the alias name, IBMSAMPL, on the System Database Directory window. The left icon indicates that the database is local, and the right icon indicates that the database is in a backup pending state.

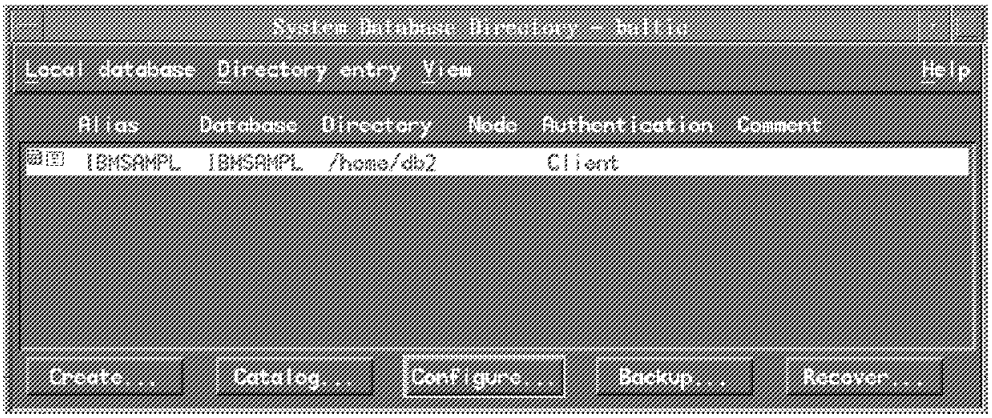


Figure 285. DB2/6000 Version 1 System Database Directory Window

4. Take an offline backup as described in 13.5.1, “Full Offline Backup and Recovery” on page 424.

Your database configuration is now ready to use both the online backup/recovery utility and the archive/retrieve log facility with ADSM.

13.5.2.2 Online Backup Using the ADSM Command Line

This example shows an online backup using the ADSM command line.

Log in to AIX as the database administrator and start the backup operation using the command:

```
db2 backup database ibmsampl online use adsm
```

Wait until a message, similar to that shown in Figure 286, appears. You should write down the time stamp as it is used to reference this backup.

```
Backup successful. The timestamp for this backup image is :
19940603144210.
```

Figure 286. DB2/6000 Version 1 Command Line Message for Successful Backup

The backup operation created an image file of your database, IBMSAMPL, and put it in ADSM server storage.

13.5.2.3 Online Backup Using the DB2/6000 Version 1 GUI

This example shows an online backup using the DB2/6000 Version 1 GUI.

Start the graphical DBA by entering:

```
db2adm
```

Select the System Database Directory option from the main window (not shown).

When the System Database Directory window appears (Figure 287 on page 435) select the database you want to use and click on Backup.

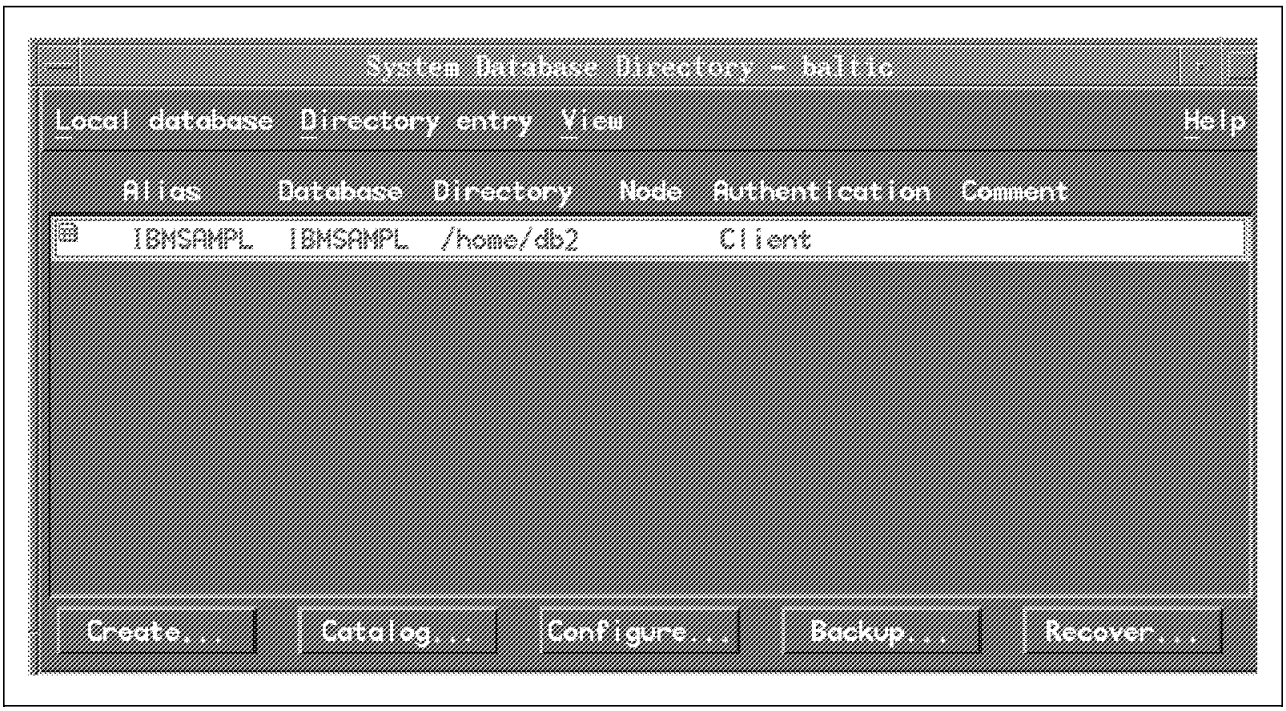


Figure 287. Select DB2/6000 Version 1 Database for Online Backup

The Backup Database window appears (Figure 288). Select ADSM as the *Backup to* option and select Online backup.

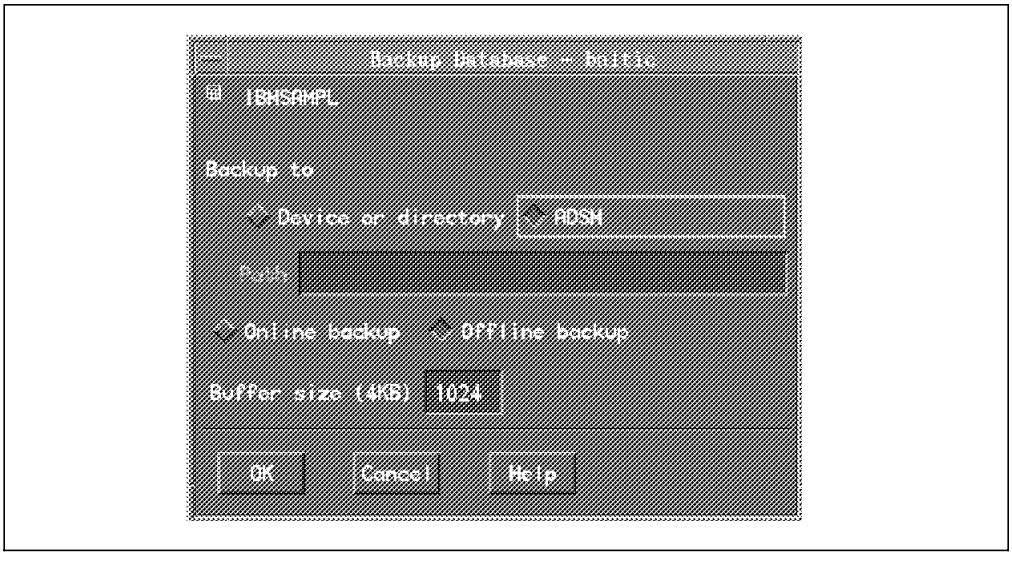


Figure 288. Select ADSM and Online Backup Options for DB2/6000 Version 1 Backup

Click on OK to execute the backup. The Backup Progress window appears (Figure 289 on page 436) to inform you of the backup state. Wait until the "The operation was successful" message appears.

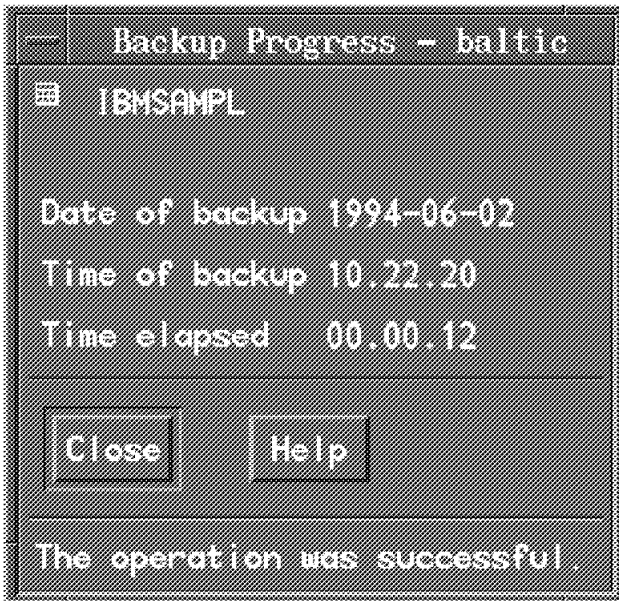


Figure 289. DB2/6000 Version 1 Online Backup Progress Window

13.5.2.4 Recovery to a Point in Time Using the Command Line

This recovery example shows the use of the command line to recover a database to a particular point in time.

Perform the database recovery as shown in 13.5.1.4, “Recovery Example Using the ADSM Command Line” on page 427.

```
db2 restore database ibmsampl use adsm taken at 19940603144210
```

Because this backup was taken while the database was online, the database manager puts the IBMSAMPL database into a *Roll Forward Pending* state. If you try to connect to it, you receive the error message shown in Figure 290.

```
$ db2 connect to ibmsampl

SQL1117N  A connection to database "IBMSAMPL" cannot be made
because of ROLL FORWARD PENDING.
```

Figure 290. DB2/6000 Version 1 Command Line Warning Message

To clear this rollforward pending state run the *Rollforward* command shown in Figure 291.

As shown in Figure 291, you receive a status message.

```
db2 rollforward database ibmsampl to 1994-06-03-13.00.00

DB20000I  The ROLLFORWARD command completed successfully.
```

Figure 291. DB2/6000 Version 1 Command Line Status Message

This command applied changes to the database to the specified point in time. You can set the following options when using roll-forward recovery:

- *Isotime*, which specifies the point in time to which all committed transactions are to be rolled forward.
- *And stop*, which specifies that the roll-forward operation is to roll back any incomplete transactions and turn off the roll-forward recovery pending state of the database.
- *End of logs*, which specifies that all committed transactions from all online archive log files be applied. The *logpath* environment variable specifies where to look for the log files.
- *Query status*, which lists the log files the database manager has rolled forward, the next archive file required, and the time stamp (in coordinated universal time) of the last committed transaction.

13.5.2.5 Recovery to a Point in Time Using the DB2/6000 GUI

This recovery example shows the use of the GUI to recover a database to a particular point in time.

Start the recovery operation using the GUI as described in 13.5.1.5, “Recovery Example Using the DB2 GUI” on page 428. When the Recover Database window appears (Figure 292 on page 438) set the following characteristics:

- Select ADSM as the *Restore from* option.
- Specify the Alias name of the database you want to restore.
- Specify the date and time to identify which backup image to use for the restore.
- Select as the *Restore to* option Existing database if you want to overwrite the existing database or New database to create a new database.
- The Buffer size parameter specifies the amount of memory to reserve as transfer buffer for the recovery operation.
- Select Roll forward if you want to make the database consistent after the recovery operation.
- Select the *To the end of online log files* option to apply all committed transactions from all online archive log files.
- Select the *To the following point in time* option to specify the point in time to which all committed transactions are to be rolled forward. In our example we specify that transactions be roll forwarded until 10:22:20 on June 3, 1994.
- Fill in the *Read log files from* option to specify where to look for log files. We do not use this option in our examples because the user exit knows the locations of the logs.

Click on OK to execute the recovery.

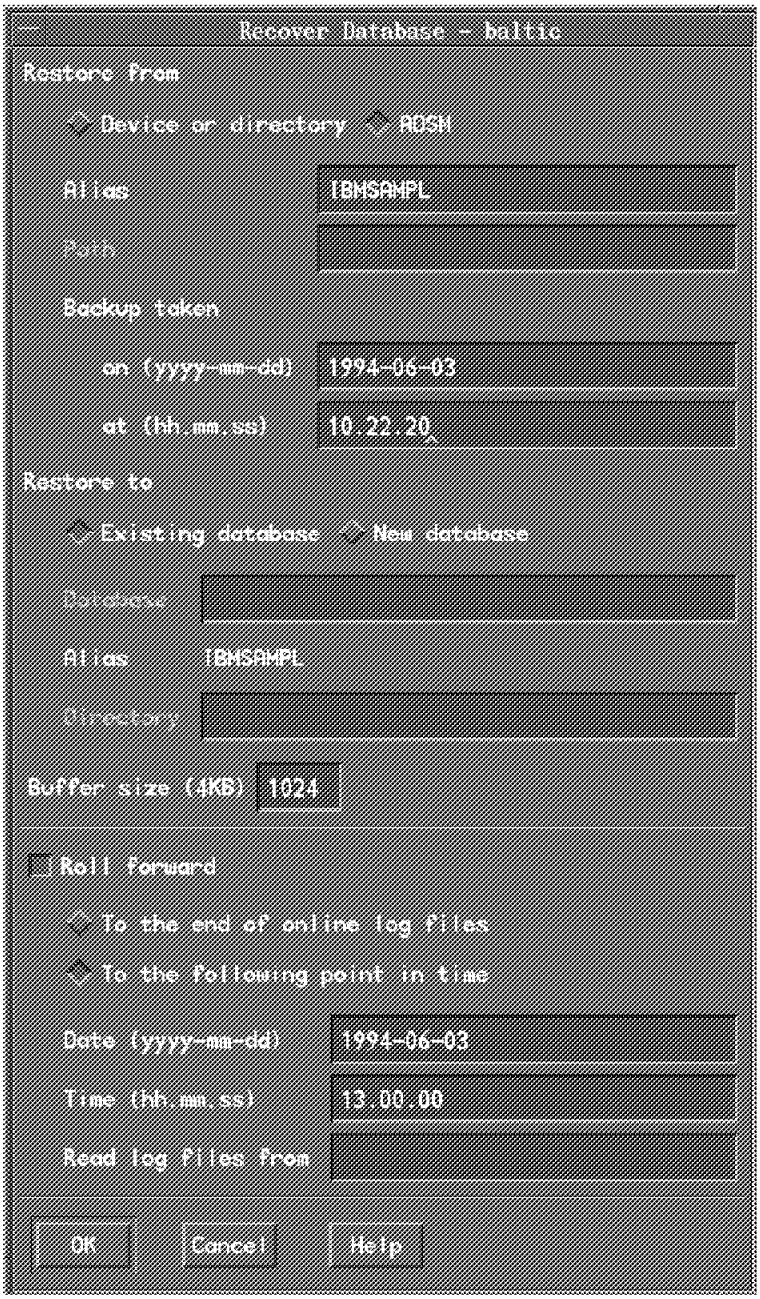


Figure 292. Select Option to Recover DB2/6000 Version 1 Database after Online Backup

A small window appears (not shown) to inform you that a database, IBMSAMPL, already exists and prompts you as to whether you want to proceed with the recovery operation. If you click on OK, the Backup Progress window appears (Figure 293 on page 439) to inform you about the recovery state and shows the progress of the recovery and roll forward. The recovery is complete when the "The operation was successful." message appears.

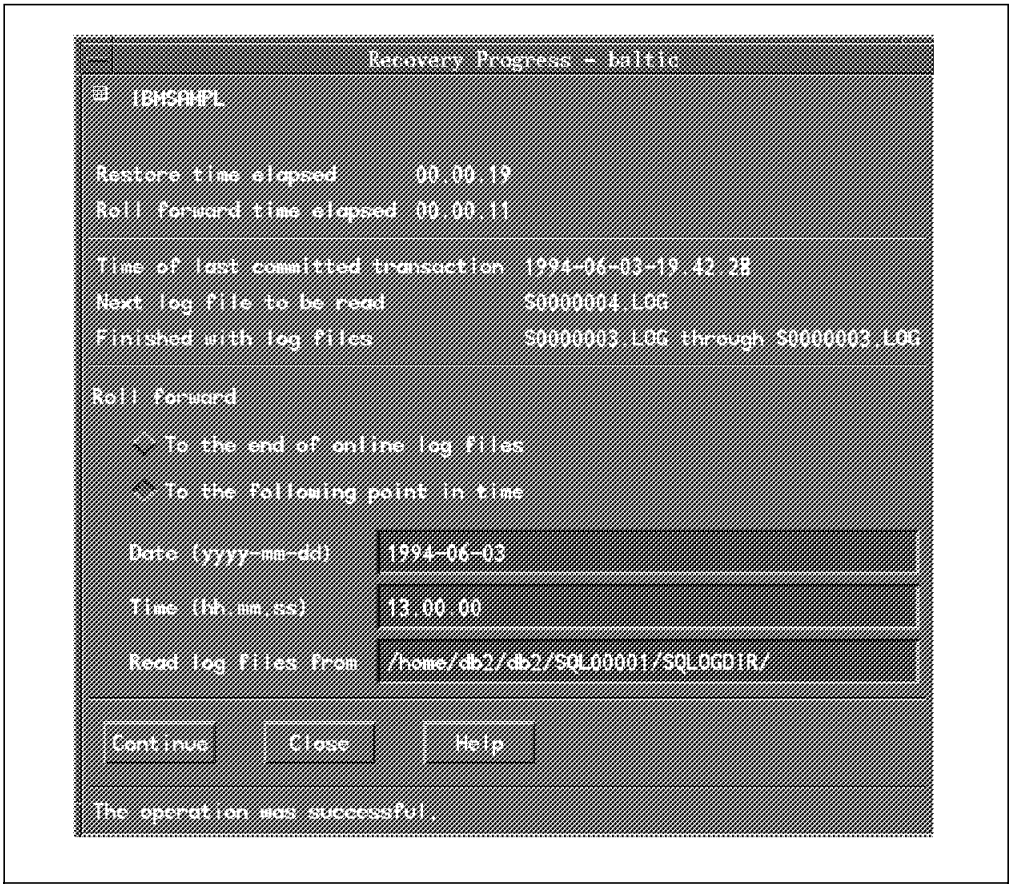


Figure 293. Recovery of DB2/6000 Version 1 Database Is Complete

13.6 Using ADSM to Back Up DB2/6000 Version 2

In this section, we examine how you can use ADSM to back up DB2/6000 Version 2 databases and tablespaces. We cover:

- Offline backup and recovery
- Online database backup and recovery
- Table space backup and recovery.

For DB2/6000 Version 2 there are some changes from DB2/6000 Version 1 we want to mention before you start. For DB2/6000 V2, the format of the GUI is changed. The database administrator utility has been replaced by the *Database Director*, which adds new concepts, such as tablespace structures, and modifies the presentation window into a tree view.

DB2/6000 Version 2 has new commands and additions to the existing command line processor that increase the number of possibilities for ADSM backup and restore for different parts of the database manager structure.

To execute a backup you need either sysadm, sysctrl, or sysmaint authority. One of these authorities is also needed if you want to restore an existing database.

13.6.1 Changing a Database to Roll-Forward Recovery

DB2/6000 online backup and recovery require the use of log files to enable roll-forward recovery. To use these log files, you must customize ADSM and activate the user exit.

The following activities enable the user exit and roll-forward recovery:

1. Complete the preparatory steps in 13.4, “Customizing the Environment to Use ADSM” on page 422.
2. Activate the user exit program.

   The user exit is an executable file that DBMS calls for archive and retrieval of log files. It enables the DBMS to interact with storage devices that are not directly supported by the operating system (such as ADSM).

   When you install DB2/6000, sample user exit programs are installed in the *$HOME/sqllib/samples/c* directory of the instance owner.

   Use the *db2uexit.cadsm* sample to archive and retrieve log files to ADSM. Perform the following steps:

   a. Log in to AIX as the instance owner.

   b. Copy the source sample file, *db2uexit.cadsm*, into your home directory by using the following command:

      ```
      cp sqllib/samples/c/db2uexit.cadsm db2uexit.c
      ```

   c. Set the *Installation Defined Variables* in the source C program, *db2uexit.c*, to suit your environment.

   d. Compile and link *db2uexit.c* with the following command:

      ```
      xlC -o db2uexit db2uexit.c -L $HOME/sqllib/adsm \
        -l ApiDS -I $HOME/sqllib/adsm
      ```

   e. As root user, copy the db2uexit file into the $HOME/sqllib/bin directory of the instance owner and the libApiDS.a library into sqllib/lib.

3. Customize the DB2/6000 environment:

   a. Log in to AIX as the *instance owner*.

   b. Start the graphical Database Director by using:

      ```
      $ db2dd
      ```

      Figure 294 on page 442 appears.

   c. Click on the arrow button next to the Database managers.

   d. Click on the arrow button next to DB26000.

   e. Click on the arrow button next to Databases.

   f. Select the database you want to enable, select *Selected* from the action bar, and click on *Configure...* in the pull-down menu.

   g. Select the *Log activity* page and enable the *Retain log files for roll-forward recovery* and *Invoke user exit for log file archiving* options.

      These two options ensure that the logs are not overwritten and that they are backed up to ADSM as soon as they become inactive.

   h. Click on OK to confirm these options.

   Now you must take an offline backup before using the database to ensure that you are using the modified environment starting from a consistent point

(the offline backup). A message window appears (not shown) to inform you to take this offline backup. Click on OK to remove it.

4. Take an offline backup as described in 13.5.1, “Full Offline Backup and Recovery” on page 424.

Your database configuration is now ready to use with ADSM and the online backup/recovery and archive/retrieve log facilities.

13.6.2 Full and Partial Offline Backup and Recovery

The DB2/6000 Version 2 backup and recovery utility can be used from either the:

- Command line interface
- Graphical user interface

  or

- Your own C, COBOL, or Fortran language program.

We do not show examples of offline or online backup and recovery using the command line as these are the same as for DB2/6000 Version 1. DB2/6000 Version 2, however, has an additional option to use multiple sessions to send data to ADSM. This option can be used to reduce the time needed to back up or recover a database.

13.6.2.1 Offline Database Backup Using Database Director

The DB2/6000 Version 2 *Database Director* can be started from the command line or by clicking on its icon. From the command line enter:

```
db2dd
```

which presents you with a window as shown in Figure 294 on page 442.

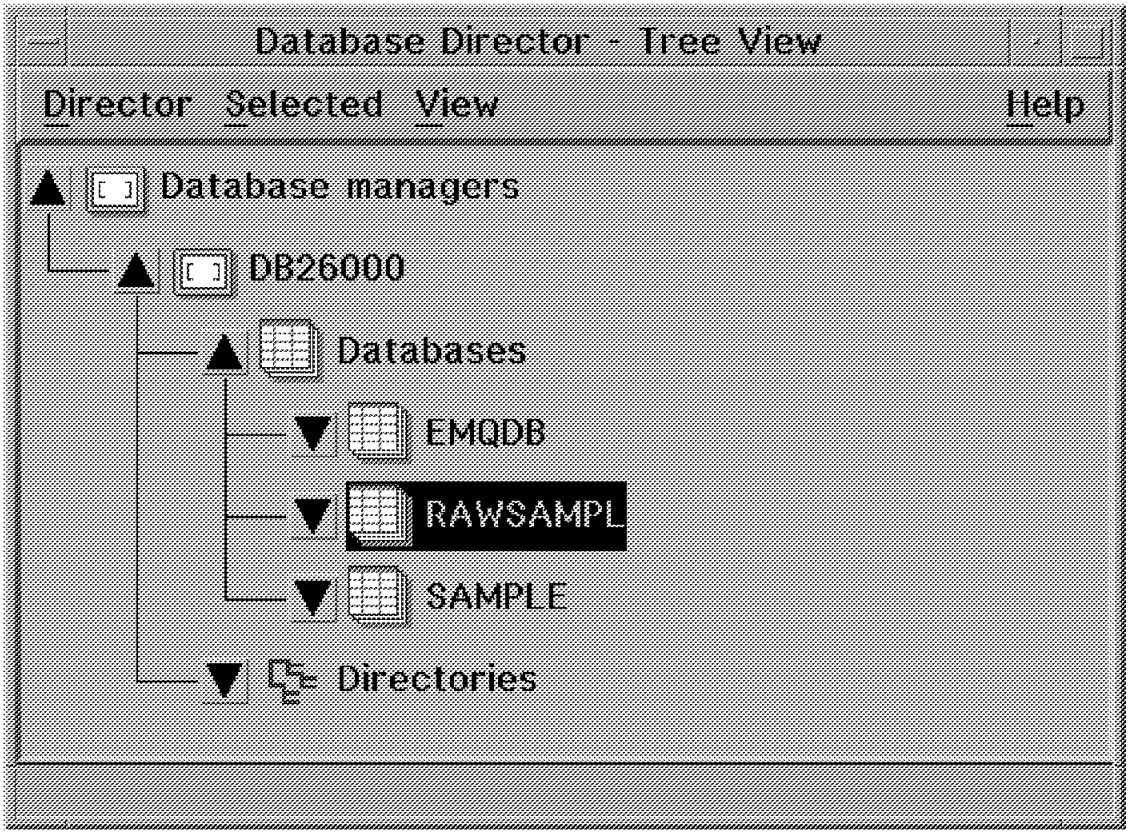


Figure 294. Select DB2/6000 Version 2 Database for Offline Backup

To start a backup of a database:

- Click on the arrow button next to Database managers.
- Click on the arrow button next to DB26000.
- Click on the arrow button next to Databases.
- Select the database you want to back up.
- Select *Selected* from the action bar and click on *Back up...* in the pull-down menu (not shown).

The RAWSAMPL-Back Up window appears (Figure 295 on page 443).

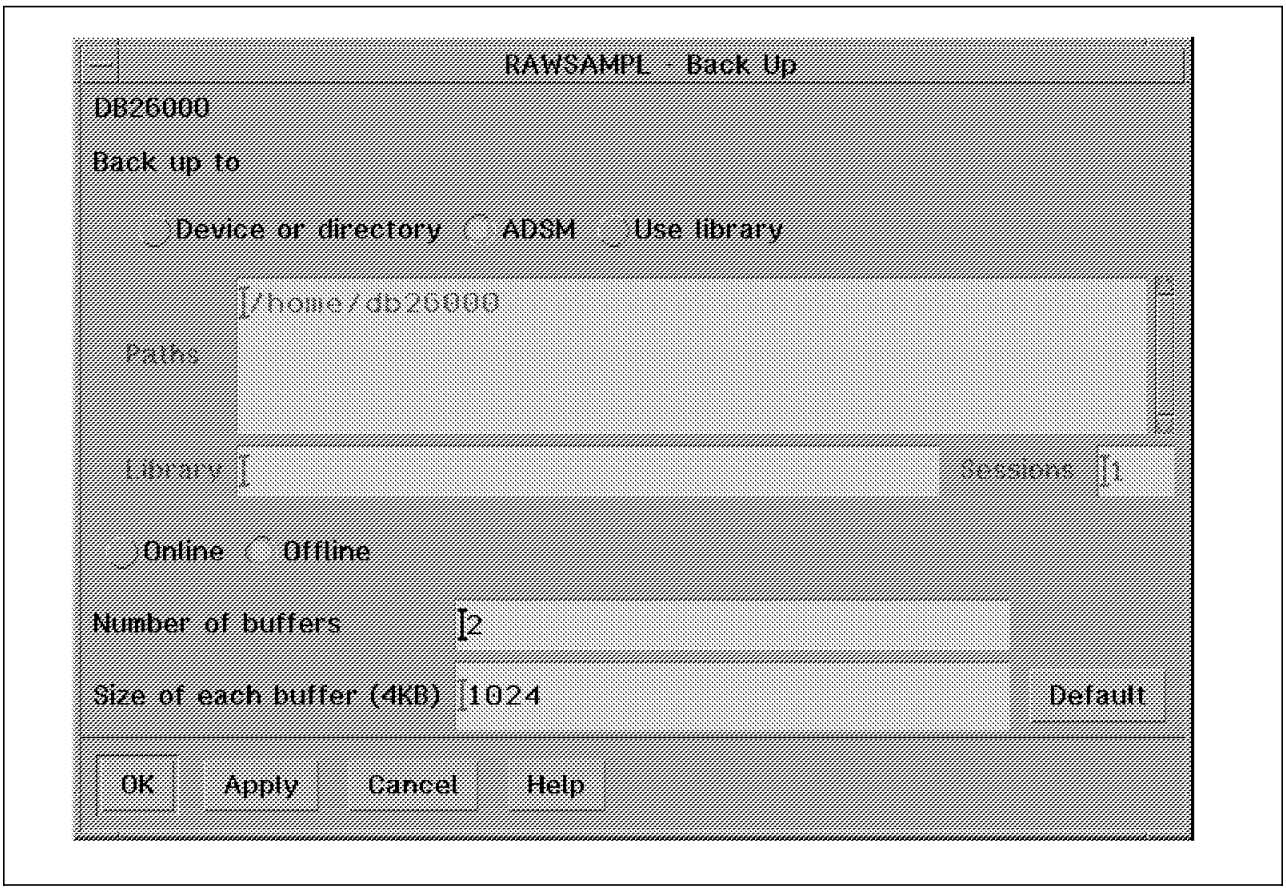


Figure 295. Select ADSM and Offline to Back Up the DB2/6000 Version 2 Database

Select the following characteristics on this window:

- ADSM as the target device
- Offline
- Number of buffers default is 2. This option tells the DBMS the number of buffers to be used.
- Click on the Default button to set the size of each buffer. The DBMS reserves memory space (buffers) for data before writing to the specified device. If you have large databases, increase the default value to improve performance during backup and recovery.

Click on OK to execute the backup. An Information window appears (Figure 296 on page 444) and informs you of a job number. Click on the OK button to close the information window.

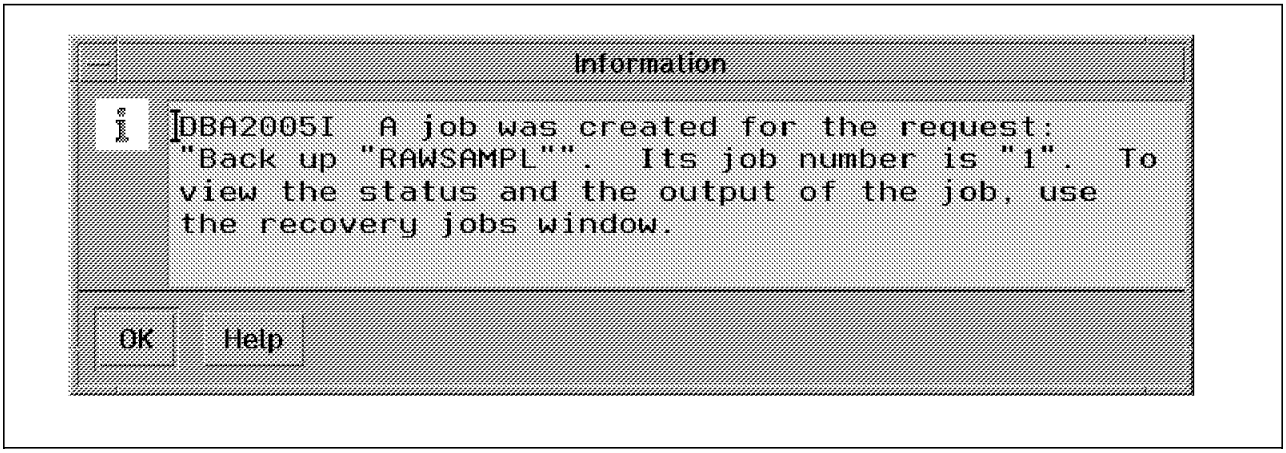


Figure 296. DB2/6000 Version 2 Information Window

13.6.2.2 Database Recovery Using the Database Director

This example shows how to recover a database using the Database Director.

First identify the time stamp of the backup. See 13.7.1, “Querying the ADSM Server for DB2/6000 Version 1 or DB2/6000 Parallel Edition” on page 456 for an example of querying the ADSM server, or alternatively use the *Recovery Jobs* utility. To use the Recovery Jobs utility enter:

```
db2jobs
```

This displays the *Jobs - Details View*, shown in Figure 297.

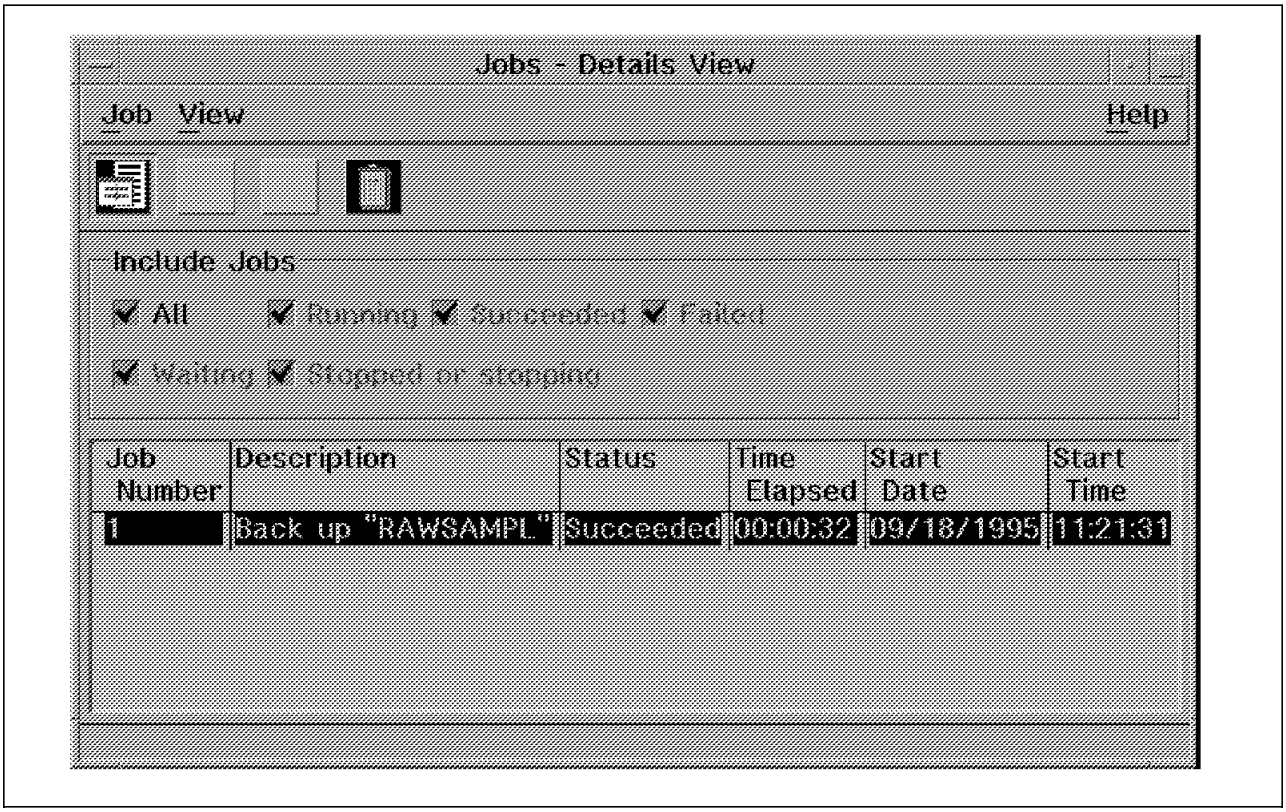


Figure 297. Select DB2/6000 Version 2 Job Number

Double-click on the desired job, and the details of the backup operation are shown in the job output window (Figure 298 on page 445).

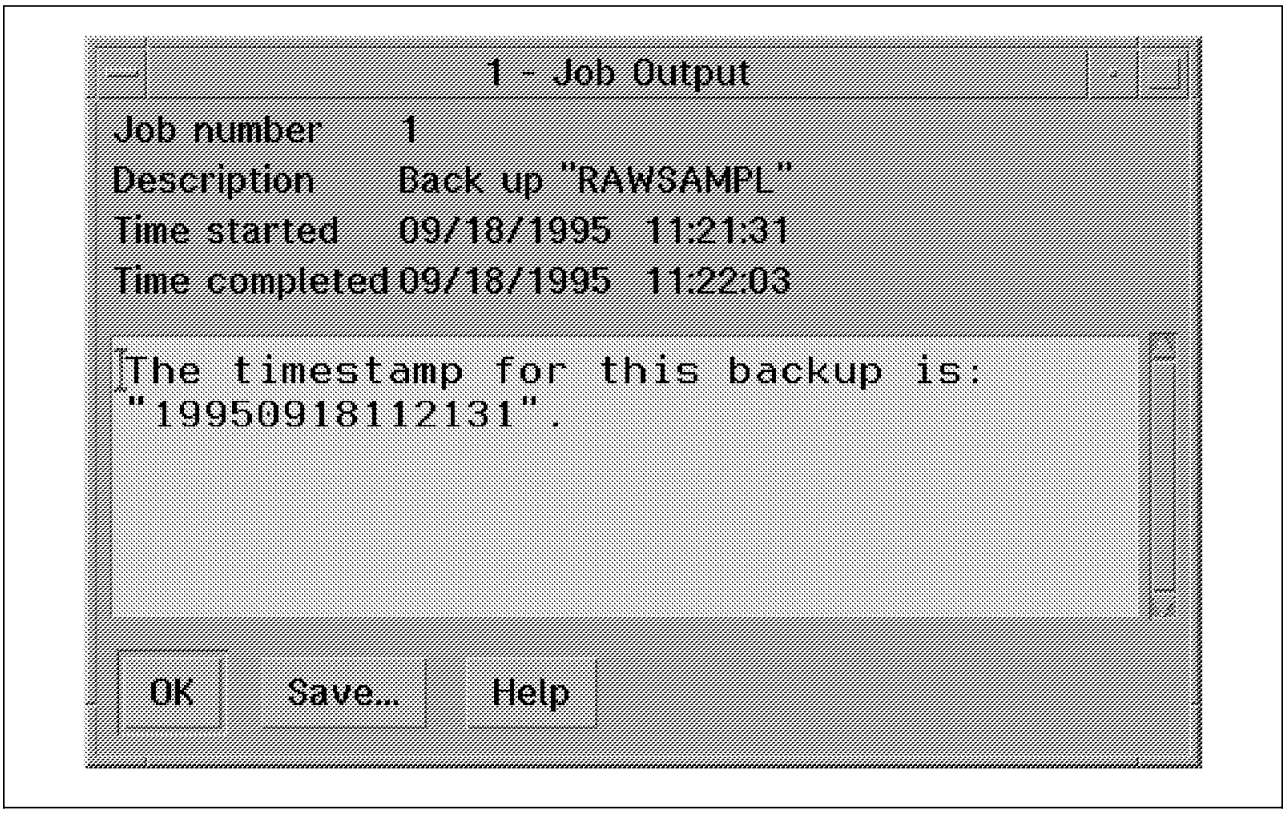


Figure 298. DB2/6000 Version 2 Job Output Window

Start the Database Director as previously described.

Select the database you want to recover to and then click on *Recover...* from the *Selected* pull-down menu. This displays the Recover Database window, shown in Figure 299 on page 446.

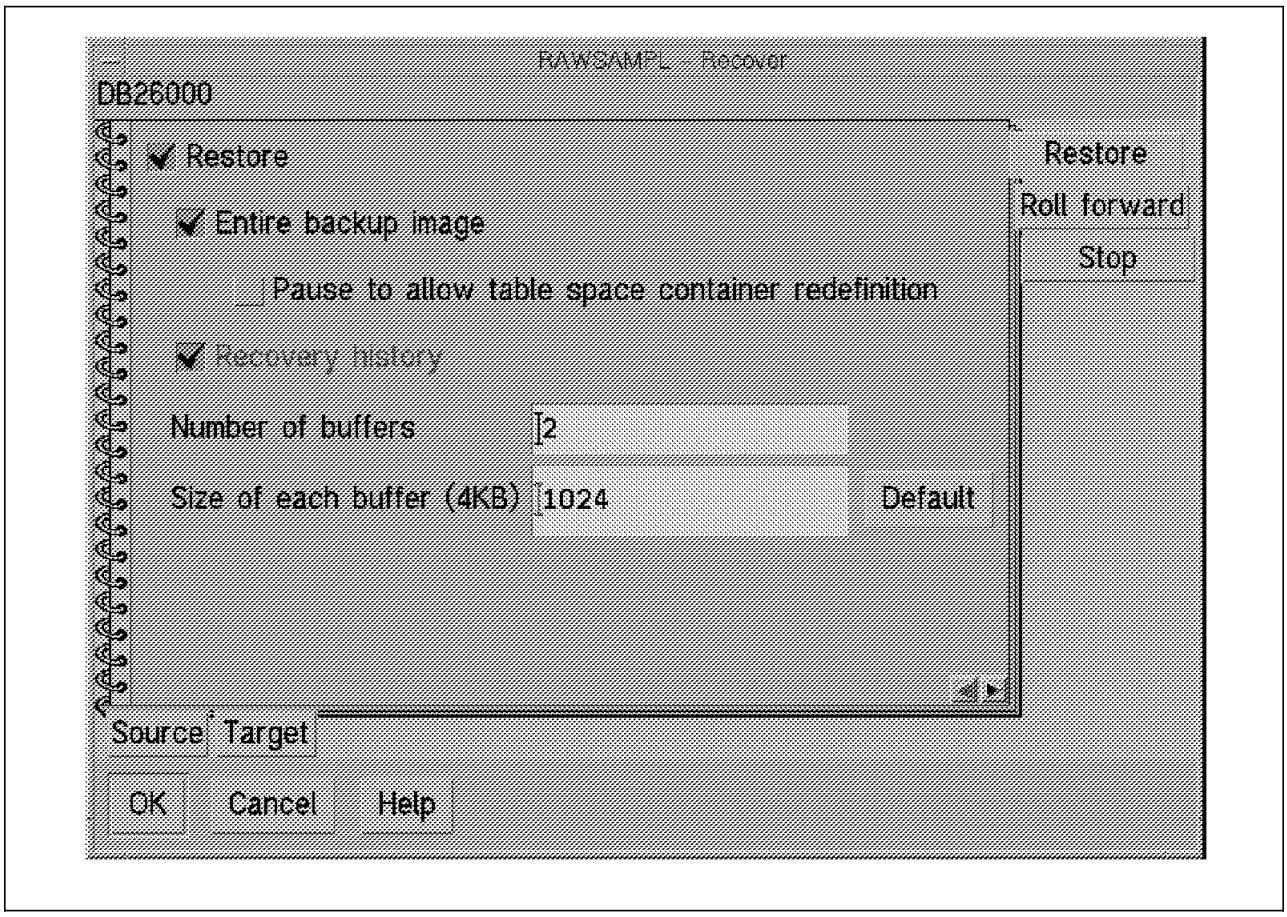


Figure 299. Page 1 (Restore Page) of the DB2/6000 Version 2 Restore Database Notebook

Set the following characteristics:

1. On the first page, click on the Default button to set the size of each buffer to 1024.
2. On the *Source* page (Figure 300 on page 447), select ADSM and enter the date and time of the backup to use for the restore.

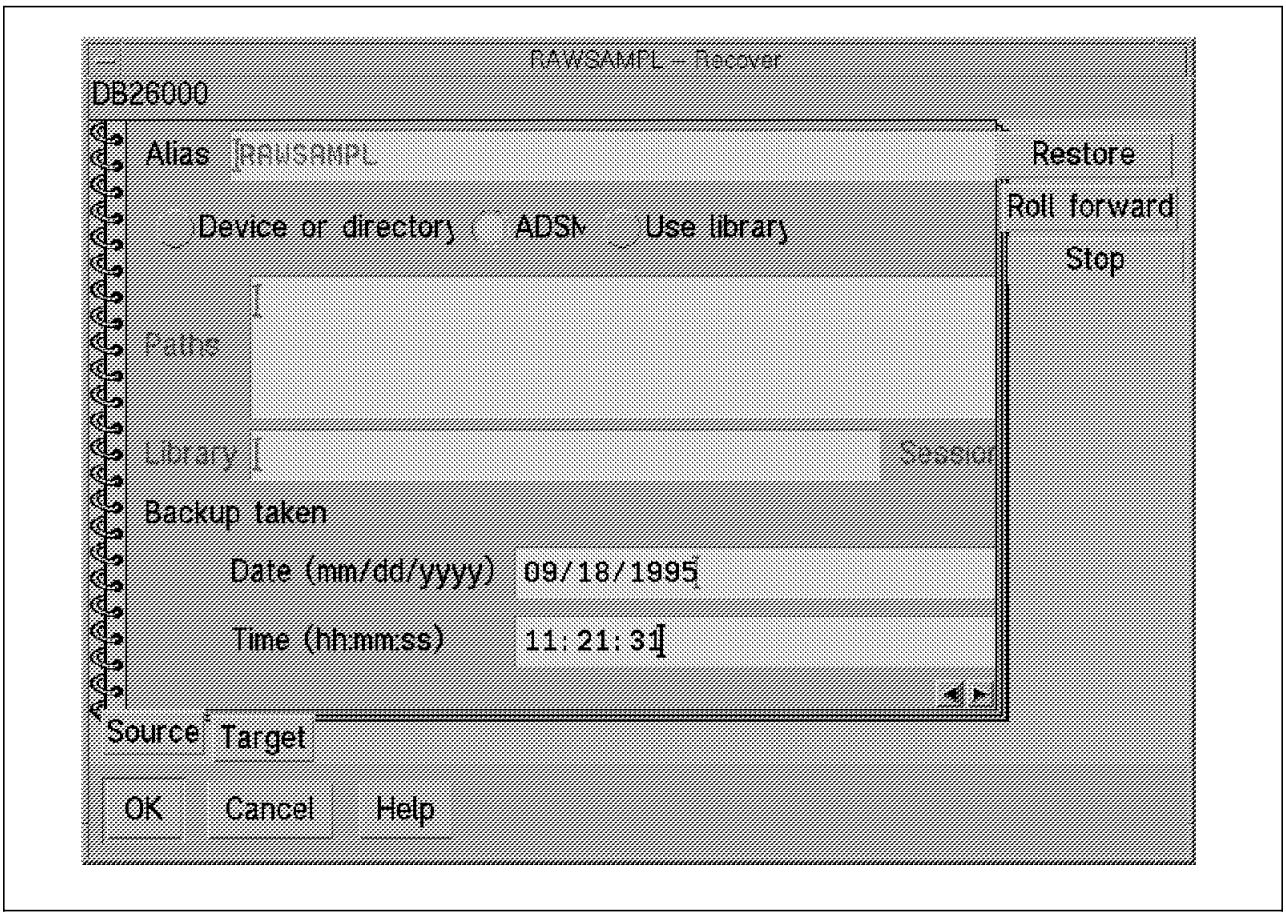


Figure 300. Page 2 (Source Page) of the DB2/6000 Version 2 Restore Database Notebook

3. On the Target page (Figure 301 on page 448), select *Existing database* if you want to overwrite the existing database or *New database* to create a new database.

   If you select *Existing database*, the alias database name is automatically filled in. If you select *New database*, enter its name (next to Database) and the directory in which you want to create the new database.

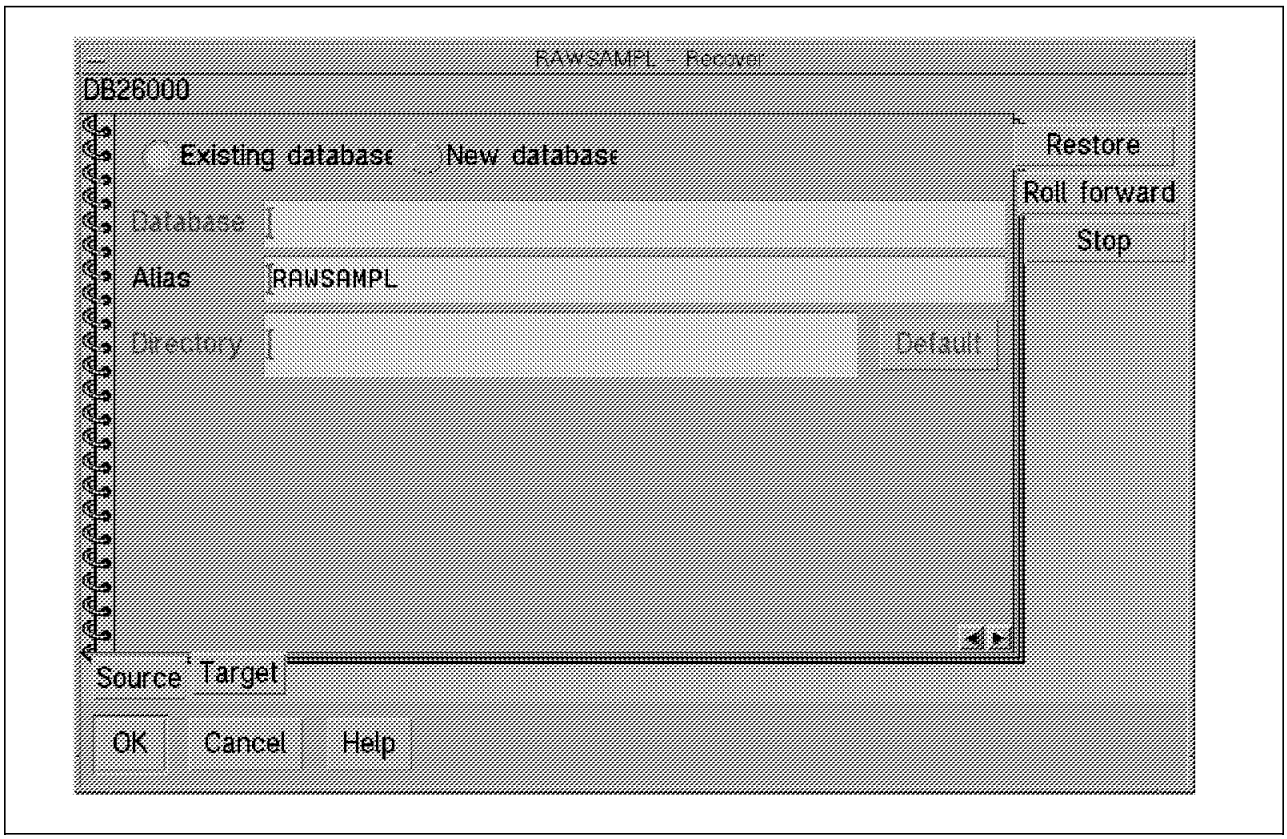


Figure 301. Page 3 (Target Page) of the DB2/6000 Version 2 Restore Database Notebook

4. On the Stop page (Figure 302 on page 449), check whether you want to make the database ready for use after recovering it.

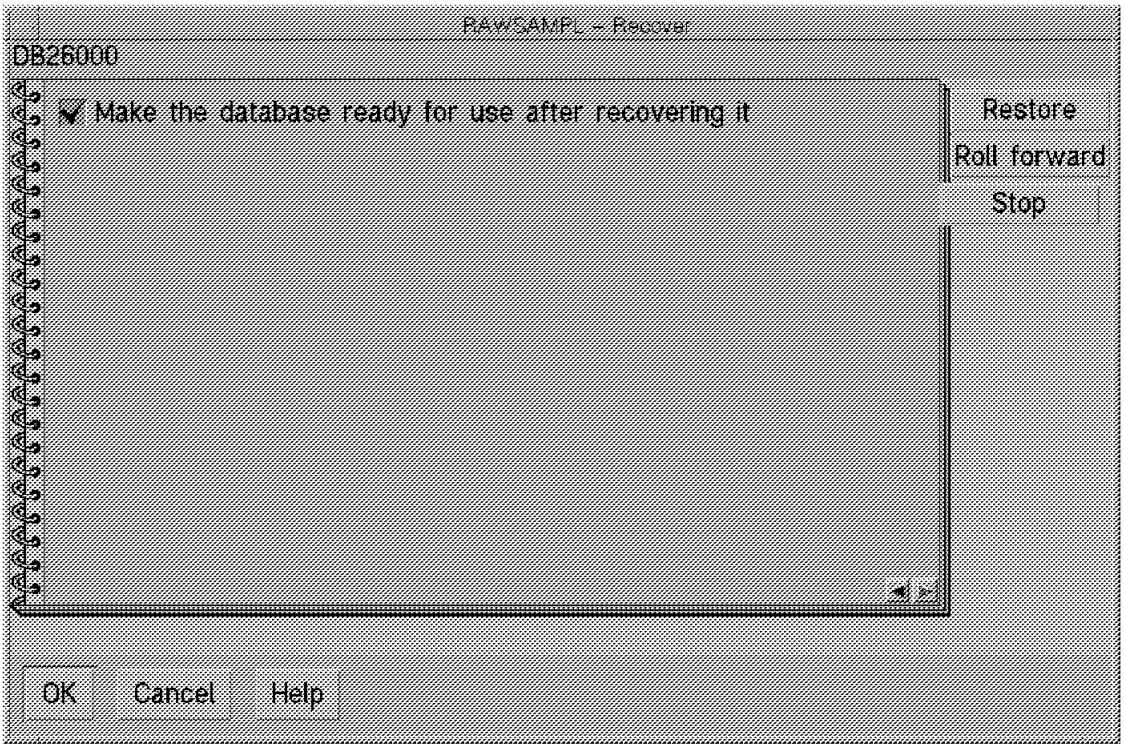


Figure 302. Page 5 (Stop Page) of the DB2/6000 Version 2 Restore Database Notebook

Click on OK to execute the restore. An Information window (Figure 303) appears and informs you of a job number. Click on OK to clear this window.

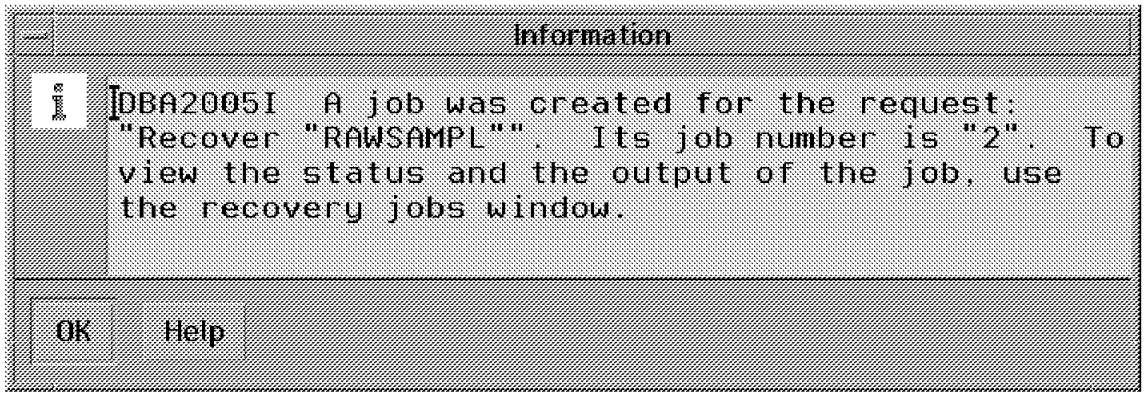


Figure 303. DB2/6000 Version 2 Information Window

To find out if your recovery job has completed successfully, go to the Recovery Jobs GUI and double-click on the desired job (Figure 304 on page 450).

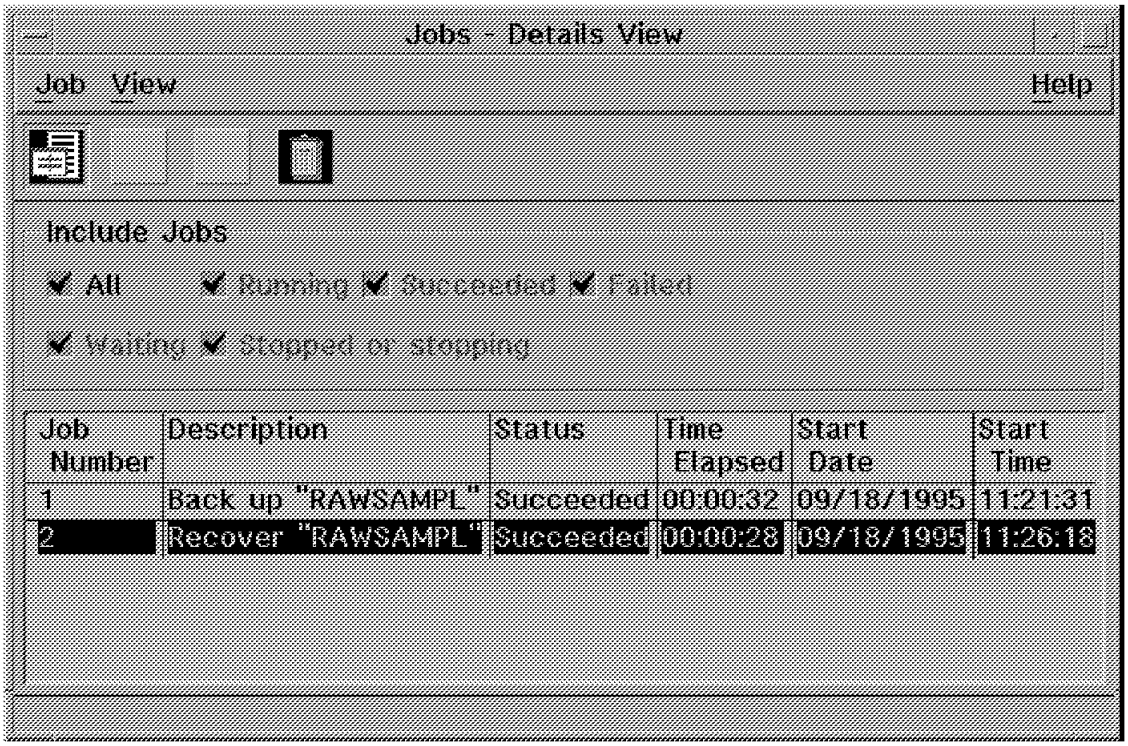


Figure 304. Double-Click on the DB2/6000 Version 2 Restore Job

A job output window is displayed (Figure 305) informing you that the database named RAWSAMPL already exists and that the files will be deleted.

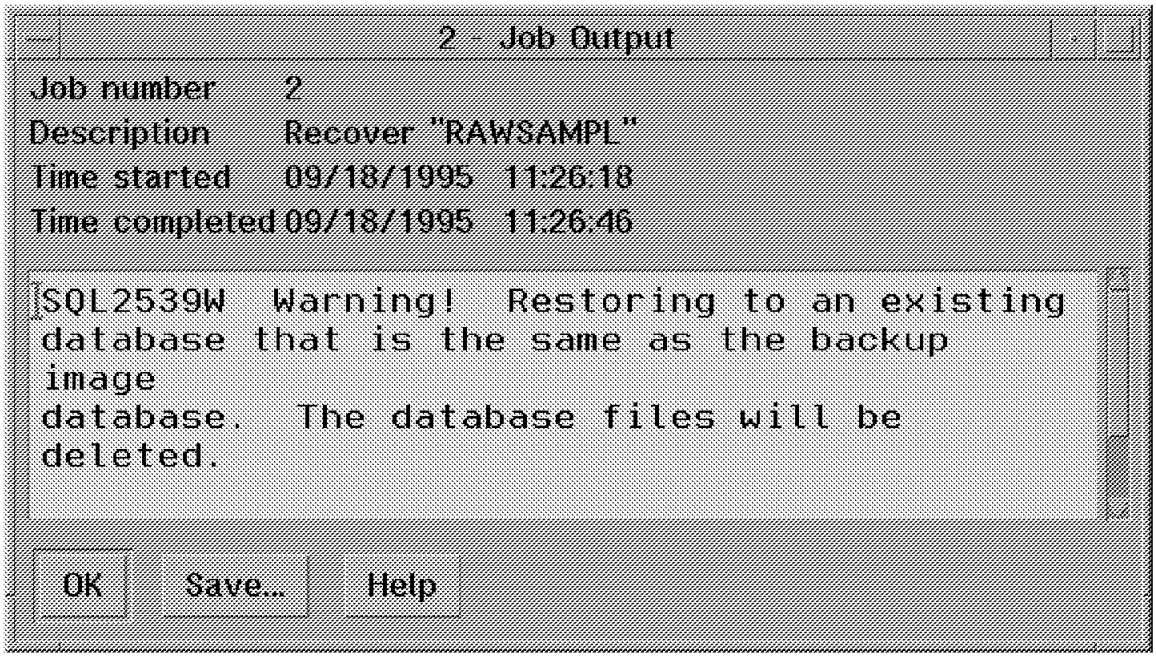


Figure 305. Restore of the DB2/6000 Version 2 Database after an Offline Backup

13.6.2.3 Offline Tablespace Backup Using Database Director

DB2/6000 Version 2 allows you to partition your database into table spaces. In this section we describe how you can back up selected table spaces.

Note: Table space level backups require that your database is in roll-forward mode. To enable roll-forward mode, see 13.6.1, “Changing a Database to Roll-Forward Recovery” on page 440.

To use the command line to back up a particular tablespace, use the **tablespace** option followed by the name of the tablespace you want to back up. You can back up multiple tablespaces with one command.

To back up tablespaces, start the Database Director and:

1. Click on the arrow button next to the Database managers (not shown).
2. Click on the arrow button next to DB26000 (not shown).
3. Click on the arrow button next to Databases (not shown).
4. Click on the arrow button next to the database containing the tablespace to back up and then click on the arrow button next to Table spaces (not shown).
5. Select all table spaces you want to back up. You can select more than one table space by holding down the *Ctrl* key and clicking on the desired table spaces (not shown).
6. Click with the right mouse button on a selected table space to open a pop-up window as shown in Figure 306 on page 452.

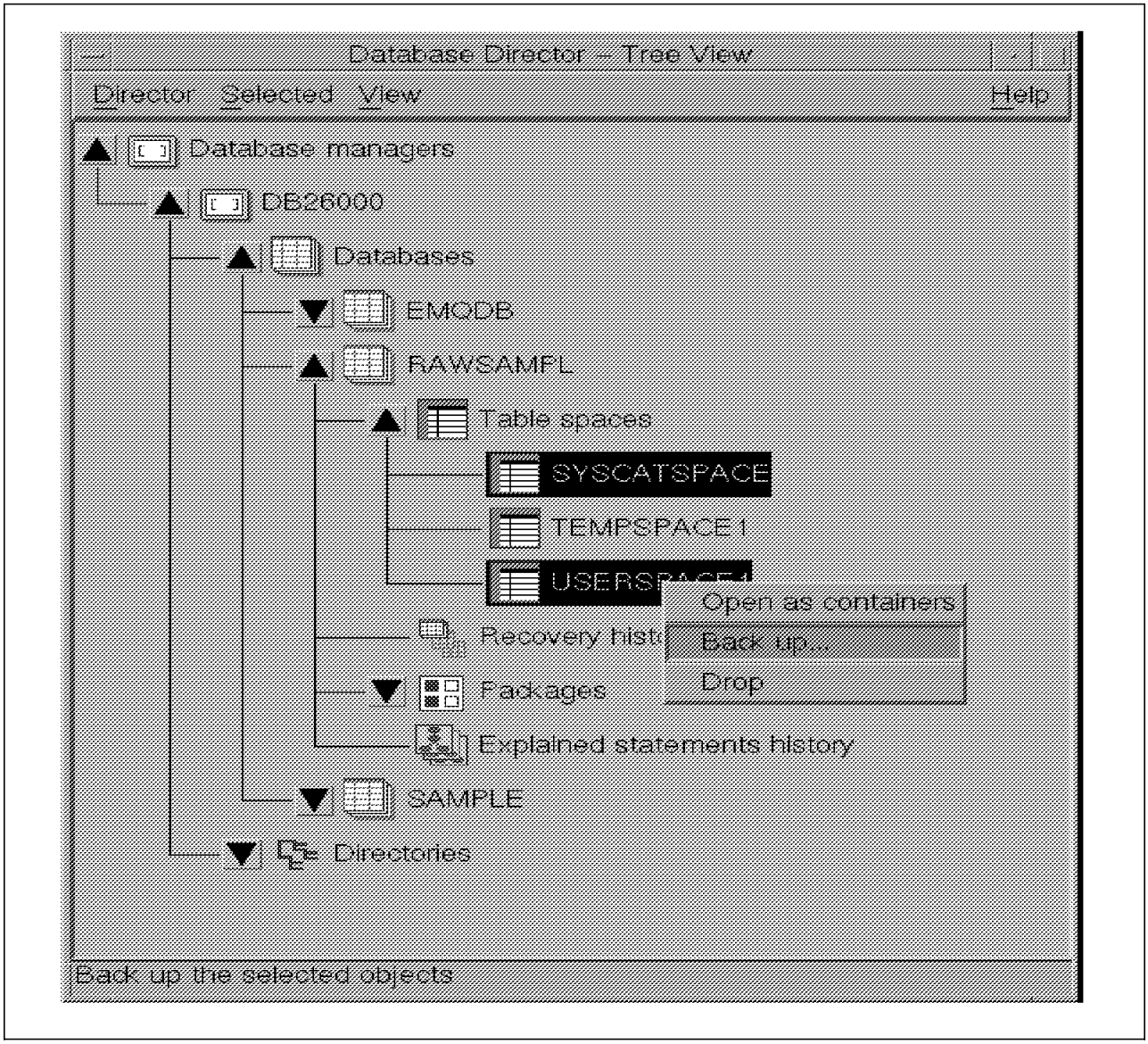


Figure 306. Select DB2/6000 Version 2 Table Spaces for Offline Backup

7. Select the *Back up...* option to display the Back Up window.

The remainder of the backup process is the same as described in 13.6.2.1, “Offline Database Backup Using Database Director” on page 441. Complete the window and click on OK to start the process.

13.6.2.4 Table Space Recovery Using Database Director

To perform a table space recovery using the Database Director you must identify the time stamp of the backup. See 13.7.1, “Querying the ADSM Server for DB2/6000 Version 1 or DB2/6000 Parallel Edition” on page 456 for an example of querying the ADSM server or 13.6.2.2, “Database Recovery Using the Database Director” on page 444 for an example of using the db2jobs GUI.

To perform a table space recovery you must have the database set to roll-forward recovery. For information on how to set roll-forward recovery, see 13.6.1, “Changing a Database to Roll-Forward Recovery” on page 440.

Note: Although table space recovery requires the database to have roll-forward recovery enabled, it does not automatically retrieve the logs from ADSM. You must retrieve the logs manually *before* you start the table space restore.

To recover using the command line specify the *tablespace* option.

Start the recovery operation using the Database Director GUI. Click with the right mouse button on *Table spaces* to open a pop-up menu (See Figure 307 on page 454).

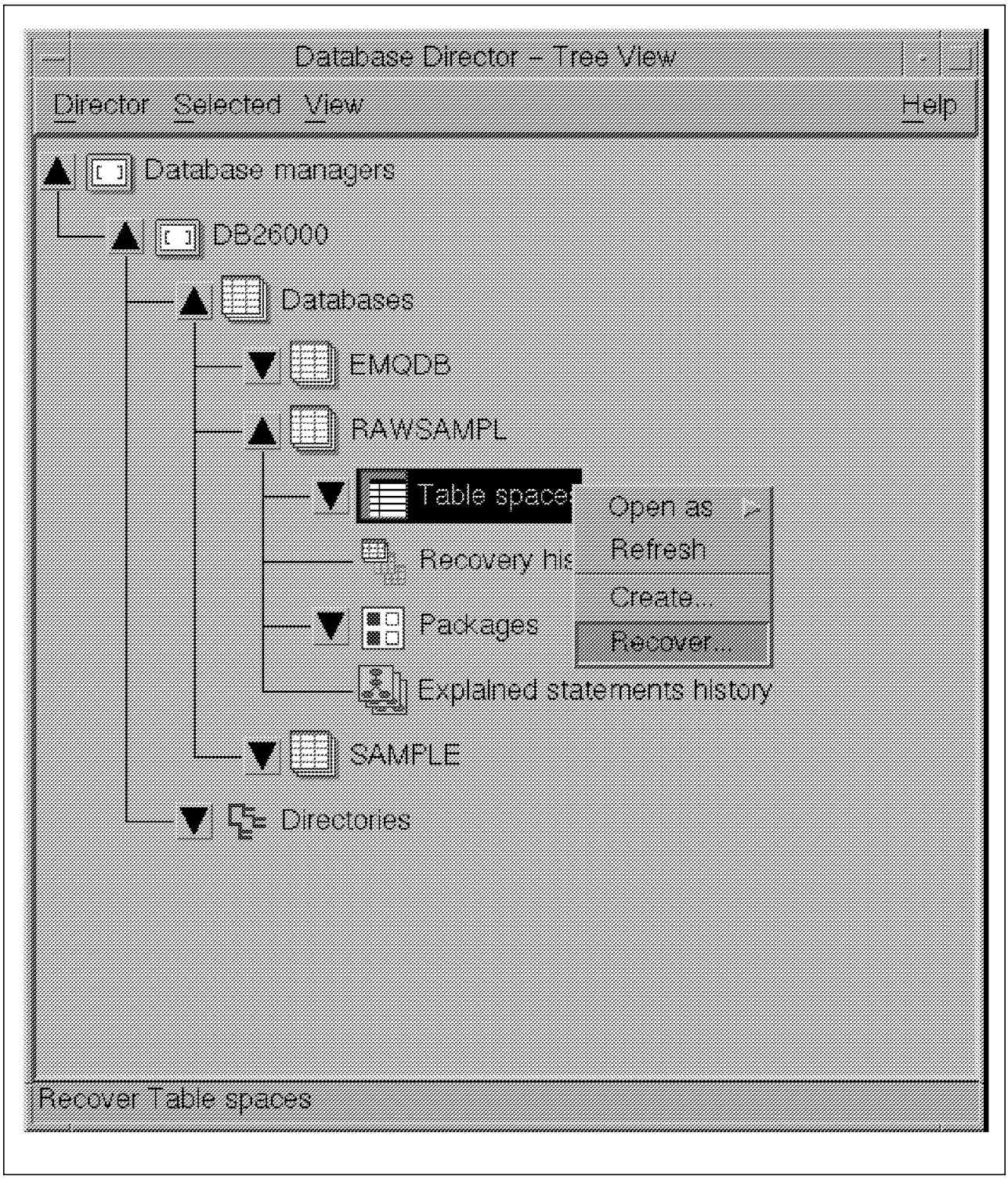


Figure 307. Select DB2/6000 Version 2 Table Spaces to Recover

Select the *Recover...* option to display the *Table Spaces - Recover* notebook (similar to Figure 299 on page 446).

Select the same options as described in 13.6.2.2, “Database Recovery Using the Database Director” on page 444 but also select the *Roll forward* option as shown in Figure 308 on page 455.

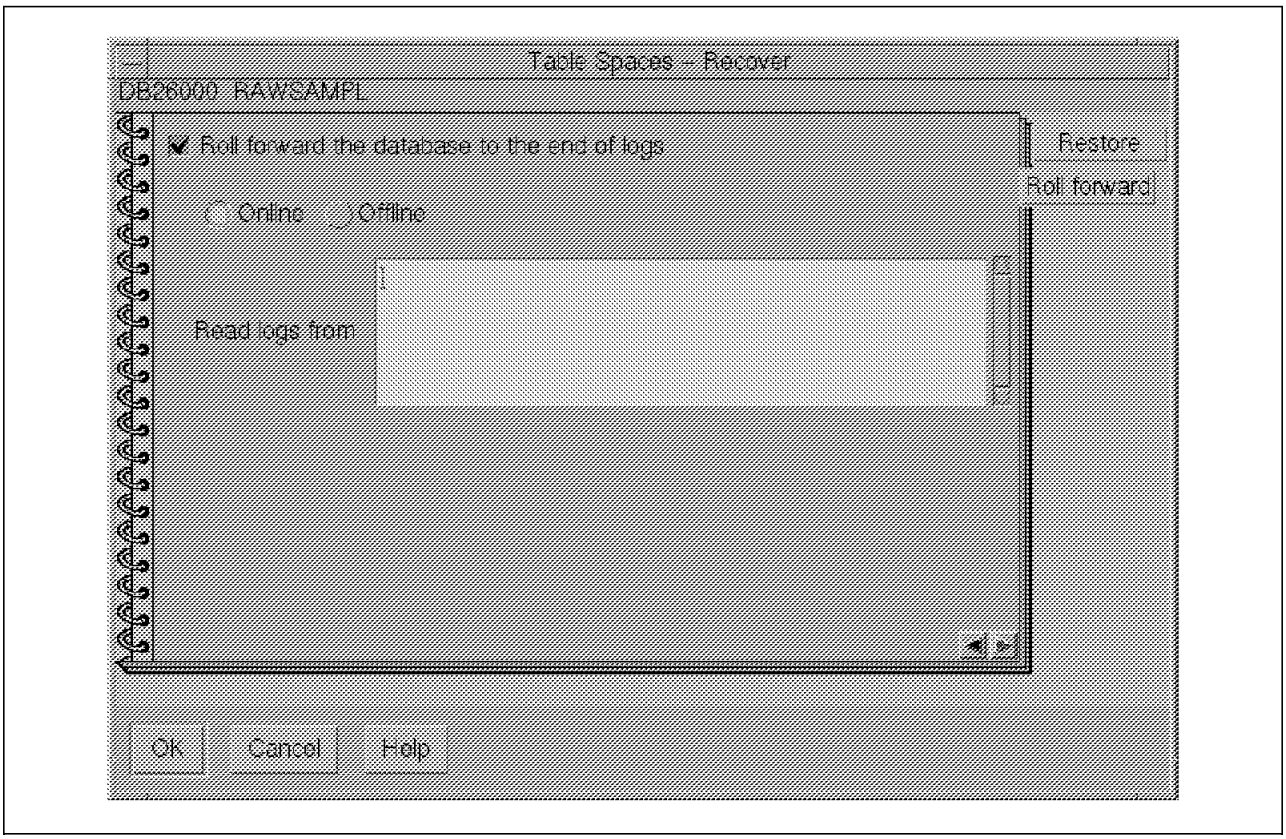


Figure 308. Page 3 of DB2/6000 Version 2 Table Spaces - Recover Notebook

The Roll forward option applies the log files after the table space has been successfully restored. Table spaces must always be rolled forward to the end of the log to synchronize them with the rest of the database. If you select Roll forward, you can choose whether the backup is restored online or not.

Click on OK to execute the restore. An Information window (not shown) appears and informs you of a job number. Click on the OK button to close the information window.

13.6.3 Full and Partial Online Backup and Recovery

DB2/6000 online backup and recovery require the use of log files to enable roll-forward recovery. See 13.6.1, “Changing a Database to Roll-Forward Recovery” on page 440, which shows how to customize the user exit and set the roll-forward option to on.

You perform online backups with DB2/6000 Version 2 using the same techniques as for offline backups except that you select *Online* instead of *Offline*. We do not show any examples of this type of backup.

13.7 Additional Considerations

Let us briefly look at how to query your ADSM server, some miscellaneous considerations, how to delete backups and archives from ADSM using the *db2adutl* utility, and DB2/6000 Parallel Edition (DB2PE) backup and restore.

13.7.1 Querying the ADSM Server for DB2/6000 Version 1 or DB2/6000 Parallel Edition

When you install DB2/6000, a sample program for querying files on the ADSM server is installed in the *$HOME/sqllib/adsm* directory of the instance owner. This utility is also included with DB2/6000 Version 2; however, we highly recommend that you use the *db2adutl* utility described in section 13.7.2, "Managing ADSM Objects Created with DB2/6000 Version 2" on page 457. The **adsmqry.csmp** sample program is coded using the C language. To use the sample program:

1. Log in to AIX as the *instance owner*.
2. Copy the source sample file, *adsmqry.csmp*, into your home directory with the following command:

   ```
   $ cp sqllib/adsm/adsmqry.csmp adsmqry.c
   ```

3. Compile and link *adsmqry.c* with the following command:

   ```
   $ cc -o adsmqry adsmqry.c -L $HOME/sqllib/adsm \
   -l ApiDS -I $HOME/sqllib/adsm
   ```

4. Run the command:

   ```
   $ adsmqry ibmsampl
   ```

The output from the query, as shown in Figure 309, includes:

- The **time stamp** of the database to use in the recovery utility.
- The **oldest log file** when the backup was taken.
- The **log files** that the DBMS moved using the user exit program.

```
Query for database SAMPLE
 Backup image 1: TIMESTAMP=19951005144210, oldestLogFile=S0000003.LOG
 Backup image 2: TIMESTAMP=19951005113242, oldestLogFile=S0000000.LOG
 Log file S0000000.LOG
 Log file S0000001.LOG
 Log file S0000002.LOG
 Log file S0000003.LOG
```

Figure 309. Sample Output from adsmqry When DB2/6000 Database Name Is Specified

If you do not specify a database name, the query will report all ADSM file spaces available. As shown in Figure 310 on page 457, two database reports are listed.

```
Query for database SAMPLE
 Backup image 1: TIMESTAMP=19951005144210, oldestLogFile=S0000003.LOG
 Backup image 2: TIMESTAMP=19951005113242, oldestLogFile=S0000000.LOG
 Log file S0000000.LOG
 Log file S0000001.LOG
 Log file S0000002.LOG
 Log file S0000003.LOG

Query for database RAWSAMPL
   Backup image 1: TIMESTAMP=19951005081603, oldestLogFile=S0000000.LOG
   No log files found
```

Figure 310. Sample Output from adsmqry When No DB2/6000 Database Name Is Specified

13.7.2 Managing ADSM Objects Created with DB2/6000 Version 2

DB2/6000 Version 2 comes with a new utility that simplifies the management of ADSM objects created by DB2/6000 Version 2. This utility, *db2adutl*, is located in the /usr/lpp/db2_*/adsm directory.

Db2adutl provides the ability to query, extract, and delete backups and log archives from ADSM. With DB2/6000 Version 2, the naming convention used for the backup files has changed. Now all backup file names include a time stamp and therefore are unique for every backup to ADSM. The backup copies in ADSM are therefore always active and never deleted by normal ADSM file expiration processing. Db2adutl provides a way of deleting backups and log files that are no longer required. To automate the management of DB2 objects in ADSM, you can run db2adutl as a cron job. For example, you can maintain only a certain number of generations of backups by using the *KEEP n* parameter on the *DELETE* option.

Figure 311 shows the syntax to invoke the db2adutl utility.

```
Usage: db2adutl
      { QUERY   [ TABLESPACE | FULL | LOGS ] |
        EXTRACT [ [ TABLESPACE | FULL ] [ TAKEN AT <timestamp> ] | [ LOGS ] ] |
        DELETE  [ [ TABLESPACE | FULL ] [ KEEP n |
                                          OLDER [THAN] [<timestamp> | n DAYS] |
                                          TAKEN AT <timestamp> ] | [ LOGS ] ] }
                [ {DATABASE | DB} <database name> ]
                [ WITHOUT PROMPTING ]
```

Figure 311. Syntax of DB2/6000 Version 2 db2adutl

QUERY queries either tablespace or full backup images, or log archives. If you type query only, it returns all three. This query is similar to the adsmqry program.

EXTRACT, if you do not qualify it, displays the name of each backup image and prompts you to extract the image. If you qualify the command you reduce the scope of the search. For this option, time stamps do not apply to log archives.

DELETE, if you do not qualify it, works similarly to extract, except that backup images are marked inactive and archive objects (log files) are deleted.

Note: The ADSM server can be configured to keep multiple backup versions. In this case, backup objects remain on the server even if you delete them by using db2adutl. If you do not want to keep these inactive objects, you have to configure your ADSM server to either of the following:

- Set the number of backup versions, if client data is deleted (VERDELETED), to 0

  or

- Set the length of time to retain the last remaining copy of a deleted file (RETONLY) to 0.

The *KEEP* qualifier *n* keeps only the newest *n* images.

The *OLDER* qualifier (*THAN* is optional) deletes any images older than the specified time stamp (this must be the complete time stamp string) or the specified number of days.

The *TAKEN AT <timestamp>* qualifier is the time stamp with which you want to perform the operation. This qualifier is not allowed with LOGS because LOGS have no time or date association.

The *DATABASE* qualifier reduces the search to a particular database.

The *WITHOUT PROMPTING* qualifier disables the prompt.

Figure 312 on page 459 shows an example of how to use the db2adutl utility.

```
>db2 backup database rawsampl use adsm

Backup successful. The timestamp for this backup image is : 19950929130942

>db2adutl query

Query for database RAWSAMPL

Retrieving full database backup information.
   full database backup image: 1, Time: 19950929130942, Oldest log: S0000053.LOG, Sessions used: 1
   full database backup image: 2, Time: 19950929142241, Oldest log: S0000054.LOG, Sessions used: 1

Retrieving tablespace backup information.
   tablespace backup image: 1, Time: 19950929094003, Oldest log: S0000051.LOG, Sessions used: 1
   tablespace backup image: 2, Time: 19950929093043, Oldest log: S0000050.LOG, Sessions used: 1
   tablespace backup image: 3, Time: 19950929105905, Oldest log: S0000052.LOG, Sessions used: 1

Retrieving log archive information.
   Log file: S0000050.LOG
   Log file: S0000051.LOG
   Log file: S0000052.LOG
   Log file: S0000053.LOG
   Log file: S0000054.LOG
   Log file: S0000055.LOG

>db2adutl delete full taken at 19950929130942 db rawsampl

Query for database RAWSAMPL

Retrieving full database backup information.  Please wait.
   full database backup image: RAWSAMPL.0.db26000.0.19950929130942.001

   Do you want to deactivate this backup image (Y/N)? y
     Are you sure (Y/N)? y

>db2adutl query

Query for database RAWSAMPL

Retrieving full database backup information.
   full database backup image: 2, Time: 19950929142241, Oldest log: S0000054.LOG, Sessions used: 1

Retrieving tablespace backup information.
   tablespace backup image: 1, Time: 19950929094003, Oldest log: S0000051.LOG, Sessions used: 1
   tablespace backup image: 2, Time: 19950929093043, Oldest log: S0000050.LOG, Sessions used: 1
   tablespace backup image: 3, Time: 19950929105905, Oldest log: S0000052.LOG, Sessions used: 1

Retrieving log archive information.
   Log file: S0000050.LOG
   Log file: S0000051.LOG
   Log file: S0000052.LOG
   Log file: S0000053.LOG
   Log file: S0000054.LOG
   Log file: S0000055.LOG
```

Figure 312. Sample Delete of a Database with DB2 common server db2adutl

Figure 312 shows the deletion of a full backup using its time stamp to identify which backup to delete. Notice that the utility shows you the number of the last log associated with each full or tablespace backup. You can work out from this log number which of the archived logs and table space backups are no longer needed or usable. For example, the last logical log for the deleted backup is 53. The last log for the remaining full backup is 54 and its time stamp shows that it

was taken at 14:22:41 on 09/29/1995. This makes it more recent than all three table space backups; therefore logs 50 to 53 are redundant.

You will almost certainly want to keep a number of “generations” of backups. In our case, the table space backups and all of the archived logs cannot be used and therefore can be deleted.

Note: Take care when you look at this display as the db2adutl utility does not necessarily display the backups in time stamp order.

13.7.3 DB2/6000 Parallel Edition

DB2/6000 Parallel Edition (DB2PE) is a key member of the DB2 family of database engines. Parallel database technology has been under development at the Almaden Research Center and elsewhere in IBM for several years.

A parallel database must push the envelop on performance and capacity; therefore it must fully leverage the capabilities of the underlying hardware, operating system, and computing environment. DB2PE exploits a shared-nothing architecture, as shown in Figure 313 on page 461, where each node has its own processor, memory, and disk, and effectively “nothing” is shared. A shared-nothing architecture has the best scalability because most of the processing is done independently on each node, with minimal interaction among nodes.

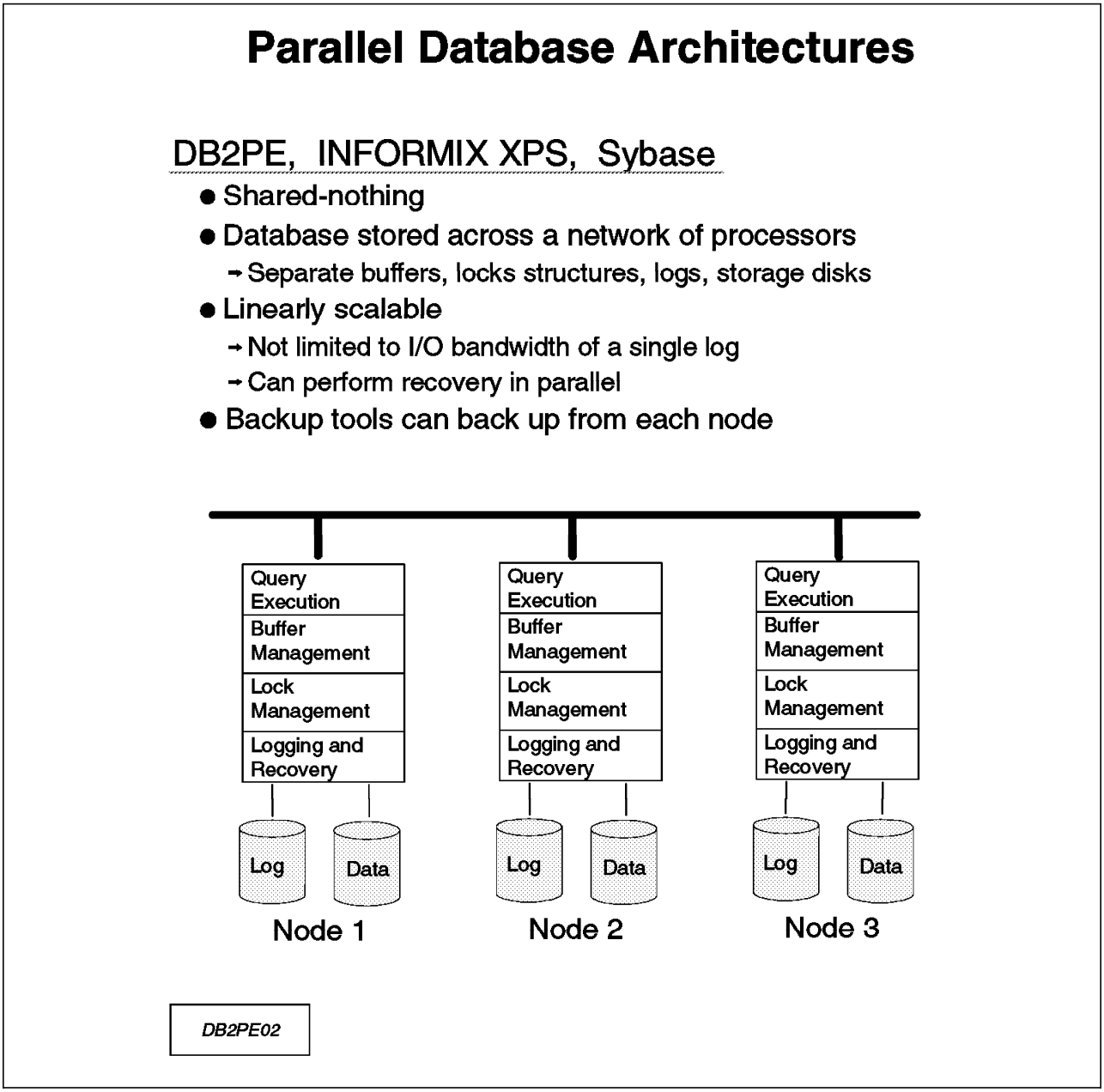


Figure 313. DB2PE Shared-Nothing Parallel Database Architecture

Each node in a DB2PE system supports a subset of the overall database. For a given user request, data access, buffer management, locking, logging, and other processing is done, where possible, independently on each node in parallel. The database appears to the application user as a single database on a single system.

DB2PE exploits the IBM POWERParallel SP2. However, it is in no way tied to or dependent on that hardware platform. DB2PE can run as multiple instances on a RISC System/6000 uniprocessor as a means of doing parallel I/O. On an SMP model of a RISC System/6000, multiple DB2PE instances can be used to leverage all processors in the single machine. DB2PE can also run on separate RISC System/6000s connected together on a LAN. Different RISC System/6000 models can be used, including a 2- or 4-way HACMP configuration.

DB2PE is based on DB2/6000 Version 1; it has the same level of function as DB2/6000, including its backup and restore functions. As described in 13.2, “DB2/6000 Backup Utilities” on page 419 and 13.5, “Using ADSM to Back Up DB2/6000 Version 1” on page 424, DB2/6000 Version 1 provides full online and offline backup and uses the ADSM API so that backups can be sent directly to ADSM.

To back up a DB2PE database with ADSM you must back up each individual DB2PE node. Just follow the steps in 13.5, “Using ADSM to Back Up DB2/6000 Version 1” on page 424. A detailed redbook on using ADSM to back up DB2PE is available, *Backup, Recovery, and Availability with DB2 Parallel Edition*, SG24-4695.

13.7.4 Miscellaneous Considerations

Here are some miscellaneous considerations regarding the use of ADSM to back up DB2/6000 databases:

- Multiple DBMS instances can coexist on the same machine. You can have an IBMSAMPL database in a test environment and an IBMSAMPL database in a production environment on the same machine, managed by different instance owners. ADSM does not recognize this distinction and therefore treats different copies of different databases made by different instance owners as the same backup object produced by the same user.

  ADSM can also be configured to keep multiple versions of a backup file. When the number of these files exceeds the configured limit, older backup versions are deleted as new backup images are created. Therefore, in a situation such as that described in the previous paragraph, the ADSM server may or may not have the backup image for the earlier backed up database. If you use the user exit to archive log files, then you may have multiple log files with the same name on your ADSM server. In this situation you cannot determine from the name of the log files to which database a particular log file belongs.

- When a client connects to an ADSM server, it establishes a *session*. If the client does not send data for a period of time specified by the *COMMTIMEOUT* parameter in the server configuration file, the session will close.

  If you receive a timeout error, consider increasing the *COMMTIMEOUT* ADSM parameter. The *COMMTIMEOUT* parameter is particularly important when issuing a redirected restore with DB2/6000 Version 2. If you use a redirected restore with DB2/6000 Version 2, the DBMS allocates all of the disk space that it is using for DMS tablespaces. Because the disk allocation can easily take several minutes, the ADSM session may time out, causing the database restore to terminate.

- The ADSM backup image naming convention has been changed in DB2/6000 Version 2 such that each backup image becomes a separate ADSM object. Therefore, ADSM versioning control is no longer used. Use the *db2adutl* utility described in section 13.7.2, “Managing ADSM Objects Created with DB2/6000 Version 2” on page 457 to manage the number of DB2/6000 Version 2 backup images you want to keep on the ADSM server.

The DB2/6000 instance is a system administrator of a DB2/6000 server process. In an AIX environment, you can have multiple DBMS servers on the same

machine to meet production and test requirements. They are completely different environments and do not share any data, databases, or instances.

Chapter 14. ADSM and DB2/2

This chapter describes in detail how to back up DB2/2 databases with ADSM. First we look at DB2/2′s DBMS structure so that you can understand which files exist and which files you have to factor in to your backup strategy. Then we look at the DB2/2 Version 1.1, 1.2, and 2.1 backup utilities. We show specific examples of how to use ADSM to back up DB2/2, including Version 1 examples, such as using the DB2/2 offline backup utility with ADSM, integrating the user exit program with ADSM, and using ADSM to directly back up the database files, and Version 2 examples where DB2/2 uses the ADSM API. We end this chapter with a look at some additional considerations, such as using the quiesce option with Version 1.2 and querying the ADSM server.

Even though this chapter focuses on DB2/2, all of the information herein is applicable to DB2 common server on all Intel platforms. The path in which DB2 common server and the ADSM Client are installed is platform specific.

14.1 DB2/2 DBMS Structure

As shown in Figure 314, the key components of a DB2/2 database that you must consider for backup and recovery are the:

- Database
- Configuration files
- Directories
- Recovery logs.

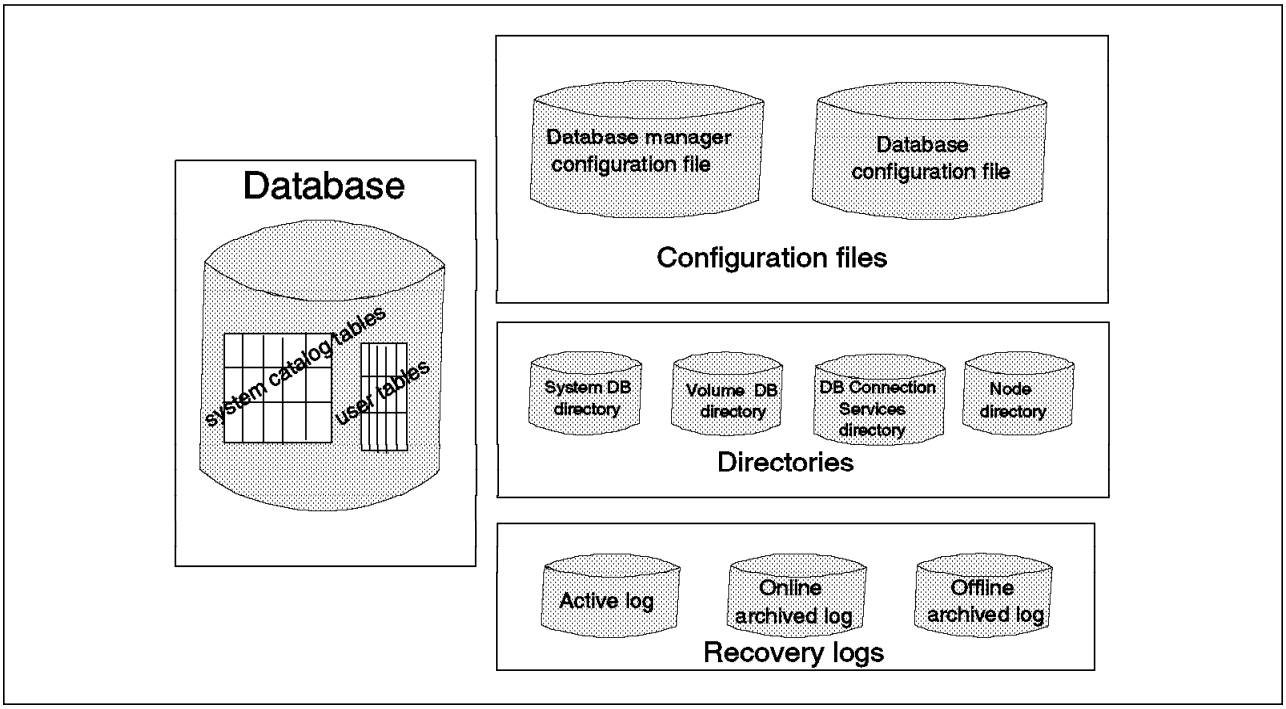


Figure 314. DB2/2 DBMS Structure

The **database** contains both control information and user data. The control information is contained in **system catalog tables**. These tables describe the logical and physical structure of the data.

The **configuration files** contain parameter values that define the resources allocated to DB2/2 and individual databases. The **database manager configuration file** contains parameter values for DB2/2 as a whole. These parameters have global applicability. The **database configuration files** contain parameter values for each individual database.

The **directories** are used to access local and remote databases. The **system database directory** identifies the name and physical location of each database. It is located on the drive on which DB2/2 resides. The **volume database directory** contains the name of a database and its physical location for a particular drive. It resides on every drive that contains a database. It is used to access the databases on that drive. The **database connection services directory** is used with another IBM product, Distributed Database Connection Services/2 (DDCS/2). It contains the host databases that your node can access. The **node directory**, or as it is sometimes referred to, the **workstation directory**, contains all remote nodes to which your node can connect.

All databases have **recovery logs** associated with them that keep records of database changes. In DB2/2, the **active logs** prevent a failure from leaving the database in an inconsistent state. These logs contain information for transactions whose changes have not yet been written to the database files. In case of a failure, the changes already made but not committed are rolled back, and all committed transactions, which may not have been physically written to disk, are redone. These actions ensure the integrity of the database. The active logs reside in the database log path directory.

Roll-forward recovery uses logs in an additional way to allow a database to be rebuilt to a specified point in time. In addition to using the information in the active logs to rebuild a database, the roll-forward function uses archived logs to reapply previous changes. When all changes in the active log are no longer needed for normal processing, the log file is closed and it becomes an archived log. It is said to be an **online archived log** when it is stored in the database log path directory. It is an **offline archived log** when it is not stored in the database log path directory. Two parameters, *new log path* and *user exit*, in the database configuration file allow you to change where the archived files are placed.

14.2 DB2/2 Backup Utilities

DB2/2 Version 1.1 provides a utility called **backup** for offline backup. Two types of backup are provided: a full backup of the **entire database** and an incremental backup of only the changed files (called **changes only**). For offline backup, no other processes can be connected to the database at the same time. Each application must be logged off. When the backup operation starts, no SQL operations are permitted.

A **user exit program** can be used in conjunction with the backup utility. It allows DB2/2 to use OS/2 commands to interact with devices (or ADSM) directly. You cannot perform a changes only backup if you enable the user exit program. The changes only backup relies on OS/2 system commands to determine which files are changed and therefore should be backed up. If you enable the user exit program, you choose other ways to save your data, such as ADSM or the copy

command. More details on the user exit program are provided in 14.3, “Using ADSM to Back Up DB2/2 Version 1” on page 468.

The **recovery** utility rebuilds a database from a previously saved database backup image. There are two recovery methods:

- **Restore recovery**, which rebuilds a database to its state when the backup copy was made. You can also use this function to duplicate a database.
- **Roll-forward recovery**, which builds a database from a restored copy, making changes to it since the time the backup was made. These changes are found in the log files. You must enable the *log retain* option using the DB2/2 configuration tool.

To use the backup utility, the database must be a local database.

DB2/2 also provides an export/import utility to move data. It can be used as a supplement to your backup strategy, but it is not a replacement for the backup utility. There is a higher risk of introducing inconsistencies into your database because no synchronization of data with logs is performed.

DB2/2 Version 1.2 provides a significant enhancement to the Version 1.1 backup utility. The DBA can perform a backup of a database with active connections, using a new **quiesce** option. This means that users, in addition to the DBA, remain connected to the database. You can think of this as somewhere in between an offline and online backup.

The quiesce option ensures that all activity on a database has been brought to a halt before a backup is taken. The backup waits until current transactions complete and prevents any new transactions from starting. Once a quiesced state has been established, a backup can be taken. After the backup is complete, user transaction activity resumes.

Two options are available with quiesce. By default, any new transactions will wait until the database is no longer quiesced. A second option is for an error message to be returned to the application after it attempts to do new work while a database is quiesced. Also, it should be noted that a user who has a running application when the backup with quiesce is issued will see the application stop and wait without receiving any kind of warning message. To an unaware user, it may appear as if the application is hung or in a loop. Activity resumes, without user intervention, immediately following the completion of the backup.

This quiesce option is available through the DB2/2 API, specifically with the *sqluback* utility, and the command line interface. Quiesce is not supported through the GUI.

DB2/2 Version 2 provides significant enhancements to the backup/restore utilities and a new load utility that is an order of magnitude faster than the existing export/import utilities. Version 2 adds the backup support already available in DB2/6000, including online backups and further integration with ADSM services by using the ADSM API. In addition, the following enhancements are new to both DB2/2 and DB2/6000 with Version 2: Databases can be partitioned into parts called **tablespaces**, and backup and recovery can be performed at the tablespace level. If a tablespace contains a single table, the backup or recovery is equivalent to a table level backup or recovery. Tablespaces can be online or offline during the backup process. During recovery, all tablespaces, other than the one being recovered, can remain online.

Another enhancement to the backup/recovery process is **parallel backup and recovery**. It is possible to perform the backup or recovery of a database or tablespace in parallel to or from multiple devices. This can drastically reduce the elapsed time requirements.

A new **load** utility significantly increases the speed of doing data loads and ensures recoverability of the data being loaded. It is intended for either bulk loading of new tables or appending large amounts of data to existing tables. The load utility is restartable and recoverable. If a failure occurs while loading data, users can continue the load without starting from the beginning. To ensure recoverability, a backup copy of the loaded table is created during the load processing.

14.3 Using ADSM to Back Up DB2/2 Version 1

In this section, we examine the ways that ADSM can be used to back up DB2/2 databases. We cover:

- Offline backup using DB2/2′s backup utility and ADSM
- Offline backup using DB2/2′s user exit integrated with ADSM
- Offline backup of the database files directly with ADSM.

We do not look at using the export/import utility because there is increased risk of introducing inconsistencies into your database with this method.

Please note that you can run DB2/2′s backup and recovery utilities from either the command line or GUI. Most of our examples in this section use the GUI. Also note that in our environment we backed up the DB2/2 server to an ADSM/6000 server. When we mention the BALTIC: AIX-RS/6000 server, we are referring to the ADSM/6000 server, not the DB2/2 server (which, from ADSM′s point of view, is an ADSM client).

14.3.1 Offline Backup Using DB2/2′s V1 Backup Utility and ADSM

This example shows the use of DB2/2′s backup utility to create a backup image of the database and ADSM commands to store that image on ADSM storage. DB2/2′s backup utility requires no other connections to the database other than the DBA that is performing the backups. This is an offline backup example.

14.3.1.1 Backup Steps

1. Start the database manager, if not already started:

   C:> startdbm

2. Ensure that NO SQL activity is running on the database you want to back up. The following command is used to obtain the status of the database to determine whether there is any SQL activity:

   C:> dbm get database status for database sample

   Figure 315 on page 469 shows the results of this command. As you can see, the *Number of current connects* is zero.

```
        Database Status


 Normalized time of last backup  = 772200786
 Time zone displacement          = -1
 Number of current connects      = 0
 Database alias                  = SAMPLE
 Database name                   = SAMPLE
 Database location               = L
 Database type                   = DB2/2
 Database drive                  = D:
```

Figure 315. Results of DB2/2 Version 1 Database Status Query

It is important that there is no other SQL activity on the database because the backup operation will try to make an exclusive connection to the database.

3. Choose the Recovery Tool icon from the main DB2/2 icon box (not shown).
4. Highlight the database you want to back up and select the Backup function.
5. As shown in Figure 316 on page 470, the DB2/2 Backup Database window appears. Select the Target Drive to which you want the backup output files to be sent (we use the *C drive*), and select the Type of Backup you want to perform. We select an *Entire Database* backup.
6. Click on Backup to start the operation.

   Two output files are created in the C:\backup directory:

   BACKUP.001

   CONTROL.001

   as well as a single file in the C:\sqluif directory:

   SQL00001.UIF

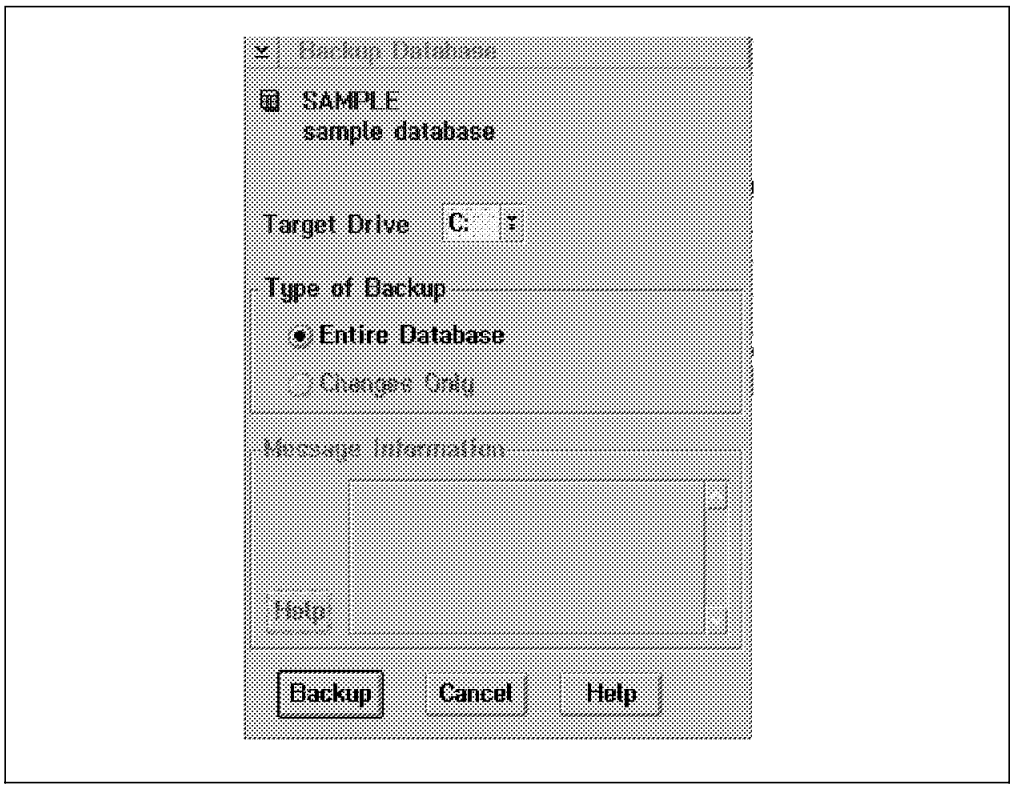


Figure 316. DB2/2 Version 1 Backup Database Window

7. A Message Information box appears to inform you of the backup state (not shown).
8. Use ADSM to back up your database image and control files.

   C:> dsmc selective c:\backup*.* -password=mars

   C:> dsmc selective c:\sqluif*.* -password=mars

   As shown in Figure 317 on page 471, several ADSM messages appear to inform you about the status of the ADSM backup.

```
Selective Backup function invoked.

 Session established with server BALTIC: AIX-RS/6000
  Server Version 1, Release 2, Level 0.0
  Server date/time: 06/22/1994 10:32:12 Last access: 06/17/1994 15:51:21

 Normal File-->         651,115 C:\BACKUP\BACKUP.001 ...................
 Normal File-->           1,954 C:\BACKUP\CONTROL.001 . Sent
 Selective Backup processing of 'C:\BACKUP\*.*' finished with no failures


 Normal File-->           1,532 C:\SQLUIF\SQL00001.UIF.................
 Selective Backup processing of 'C:\SQLUIF\*.*' finished with no failures

 Total number of objects inspected:        3
 Total number of objects backed up:        3
 Total number of objects updated:          0
 Total number of objects rebound:          0
 Total number of objects deleted:          0
 Total number of objects failed:           0
 Total number of bytes transferred:    637.8 KB
 Data transfer time:                    3.38 sec
 Data transfer rate:                  188.46 KB/sec
 Average file size:                    318.8 KB
 Elapsed processing time:            0:00:11
```

Figure 317. ADSM Selective Backup Processing Messages for DB2/2 Version 1

9. To save space, remove files in c:\backup. First run the *attrib* command to remove the *READ ONLY* attribute to the files.

   C:> attrib -r c:\backup*.*

   C:>del c:\backup*.*

   C:>del c:\sqluif*.*

14.3.1.2 Restore Steps

1. Use ADSM to restore the database image and control files.

   C:> dsmc restore c:\backup*.* -password=mars

   C:> dsmc restore c:\sqluif*.* -password=mars

   Figure 318 on page 472 appears to report on the ADSM restore processing.

```
ADSTAR Distributed Storage Manager
Command Line Backup Client Interface - Version 1, Release 1, Level 0
5648-020 (C) Copyright IBM Corporation, 1990, 1993, All Rights Reserved

Restore function invoked.

Session established with server BALTIC: AIX-RS/6000
 Server Version 1, Release 2, Level 0.0
 Server date/time: 06/22/1994 10:50:15 Last access: 06/22/1994 10:39:41

Restoring      651,115 C:\BACKUP\BACKUP.001 ..................... Done
Restoring        1,954 C:\BACKUP\CONTROL.001 . Done
Restoring        1,532 C:\SQLUIF\SQL00001.UIF....Done

Restore processing finished.
```

Figure 318. ADSM Restore Processing Messages for DB2/2 Version 1

2. Ensure that there is no SQL activity on the database you want to restore.

   C:> dbm get database status for database sample

   Figure 319 shows that there is no SQL activity because the *Number of current connects* is zero.

```
        Database Status


Normalized time of last backup  = 772200786
Time zone displacement          = -1
Number of current connects      = 0
Database alias                  = SAMPLE
Database name                   = SAMPLE
Database location               = L
Database type                   = DB2/2
Database drive                  = D:
```

Figure 319. DB2/2 Version 1 Database Status Query Messages

3. Choose the Recovery Tool icon from the main DB2/2 icon box (not shown).
4. Highlight the database you want to recover and select the Recover function. Then you decide either to overwrite the existing database or create a new one from the recovery operation. We select to recover an existing database (not shown).
5. The Recover an Existing Database window appears (Figure 320 on page 473). Select the source drive you used to store the backup; in our example, the C Drive.
6. Select that you want Roll forward recovery to occur after the database is restored.
7. Click on Start to start the operation.

Figure 320. Recover an Existing DB2/2 Version 1 Database Window: Enter Recovery Options

After the restore is complete, roll-forward recovery starts by looking for transactions present in the active log files. As shown in Figure 321 on page 474, a message information box appears to inform you about the roll-forward status.

8. Click on Resume to continue rolling forward after manually copying more online log files or changing the roll-forward parameters.
9. Click on Stop to stop rolling forward the logs, roll back any incomplete transactions, and unlock the database by removing the roll-forward pending state so that it can be used again.

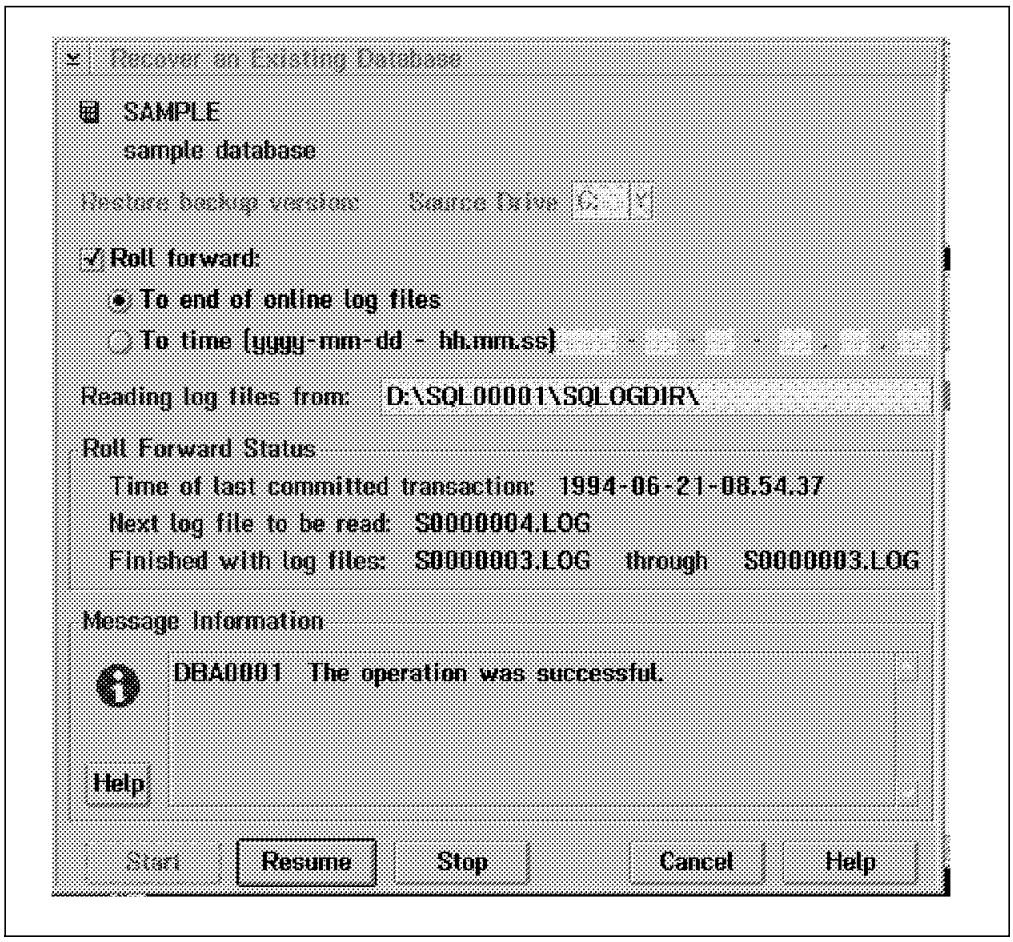


Figure 321. Recover an Existing DB2/2 Version 1 Database Window: Operation Successful

14.3.2 Offline Backup Using the DB2/2 V1 User Exit Integrated with ADSM

The DB2/2 backup and recovery utility can call a user exit program to store and retrieve database files. The user exit program allows the database manager to use OS/2 commands to interact with devices directly or not directly supported by the operating system. This example shows using the DB2/2 backup utility in conjunction with the user exit program, where the user exit program contains ADSM commands to store the database backup image files in ADSM storage.

14.3.2.1 User Exit Program Setup

When you install DB2/2, four different sample user exits are installed in the \SQLLIB directory. These help you write your own programs to meet your particular needs.

Each sample consists of four sections: backup, restore, archive, and retrieve. These sections are used by the database manager to save database files (backup and restore) and log files (archive and retrieve).

The user exit program must be named SQLUEXIT.CMD before the database manager can use it, and it must be placed in a directory present in the PATH environment variable, usually in the \SQLLIB directory. Our example uses the SQLUEXIT.EX4 sample program with OS/2 commands (such as ADSM commands) to perform the backup and restore functions.

The user exit program contains many procedures used by the backup, restore, archive, and retrieve functions called by the DBMS. These procedures verify the OS/2 environment, such as whether the directory exists and interpreting the return codes. To integrate ADSM commands with DB2/2, changes are needed to some of these procedures. Here is a sample integration you can use to implement your own user exit program.

Modify the user exit program as follows:

1. Copy SQLUEXIT.EX4 to SQLUEXIT.CMD.

   copy x:\sqllib\sqluexit.ex4 x:\sqllib\sqluexit.cmd

   where *x* is the drive on which DB2/2 is installed.

2. Modify the INIT PROCEDURE in the SQLUEXIT.CMD as shown in Figure 322.

   Add a stanza for each database you want to manage with the user exit program. You can set up a different environment for each database:

   - *DB_NAME* is the database name you want to manage with the user exit.
   - *DB_DIR* is the directory you want to use to back up your database files.
   - *DB_ARC_DIR* is the directory you want to use to archive the log files.

```
DB_NAME.1    = 'SAMPLE'
DB_DIR.1     = 'D:\DB_BACKS\SAMPLE'
DB_ARC_DIR.1 = DB_DIR.1'\ARCHIVE'

DB_NAME.2    = 'SAMPLE1'
DB_DIR.2     = 'D:\DB_BACKS\SAMPLE1'
DB_ARC_DIR.2 = DB_DIR.2'\ARCHIVE'
```

Figure 322. DB2/2 Version 1 INIT PROCEDURE Modifications

The INIT procedure also contains other variables we need to consider: AUDIT, COMPRESSION, and DESTINATION. Figure 323 shows the default values for these variables.

- Set the AUDIT option to **Y** to log each user exit function to disk.
- Set the COMPRESSION option to **Y** to invoke the PKZIP2 OS/2 utility. This produces one compressed output file containing all database files. This helps keep all database files together, making it easier to recover.
- The DESTINATION variable is not used in our example because we store the log files in ADSM storage instead. The destination variable is used to specify the target directory when archiving log files.

```
/*******************************/
/* DEFAULT CONFIGURATION VALUES */
/*******************************/
AUDIT       = 'N'
COMPRESSION = 'Y'
DESTINATION = 'C:\LOGS'
```

Figure 323. Default Values for Variables in DB2/2 Version 1 INIT Procedure

3. Modify the BACKUP PROCEDURE in SQLUEXIT.CMD as shown in Figure 324 on page 476.

The BACKUP procedure produces a compressed database image file. It is then saved on ADSM storage and deleted from the OS/2 disk. We add the ADSM dsmc command and the delete command as highlighted in Figure 324 on page 476. Note, by adding the ADSM command in this spot, the dsmc return code is managed from the checkrc function.

```
if( MODE = '2' & COMPRESSION = 'Y' ) then
  do
     'PKZIP2 'db_dir'\'DB_NAME' 'rsp_line
 'dsmc selective' db_dir'\'DB_NAME'.zip' '-password=mars'
    if rc <> 0 then
      do
        ELINE = 'Call to PKZIP2 Failed'
        sqluexitrc = rc
     end
else
     do
        'del' db_dir'\'DB_NAME'.zip'/* to save space /*
     end
  end
else
  do
    'XCOPY 'rsp_line' 'db_dir     /* XCOPY the requested file(s)*/
    call checkrc                  /* Check return code from OS/2 */
  end
```

Figure 324. DB2/2 Version 1 BACKUP Procedure Modifications

4. Modify the RESTORE PROCEDURE in SQLUEXIT.CMD as shown in Figure 325 on page 477.

   The RESTORE procedure invokes the dsmc command to retrieve the compressed database image file and then the pkunzip2 utility to uncompress it and place the database files in the correct location. Add the highlighted ADSM and pkunzip commands to the RESTORE procedure as shown in Figure 325 on page 477.

```
if( MODE = '2' & COMPRESSION = 'Y' ) then
  do
   'PKUNZIP2 'db_dir'\'DB_NAME' 'DB_DRIVE||REST_DIR
   if rc <> 0 then
     do
        ELINE = 'Call to PKUNZIP2 Failed'
        sqluexitrc = rc
     end
  end
else
 do
'dsmc restore' db_dir'\'DB_NAME'.zip' '-password=mars'
'pkunzip -o' db_dir'\'DB_NAME DB_NAME'.UIF' db_dir
     'XCOPY 'db_dir'\'filelist' 'DB_DRIVE||REST_DIR
     call checkrc    /* Check return code from OS/2's XCOPY command */
 end

 signal FINISH
```

Figure 325. DB2/2 Version 1 RESTORE Procedure Modifications

5. Modify the ARCHIVE and RETRIEVE procedures in SQLUEXIT.CMD so that ADSM is used to archive and retrieve the log files.

   The ARCHIVE and RETRIEVE procedures are not "user initiated" like backup and restore procedures. The Archive procedure is called by the DBMS when a log file becomes inactive. Archive copies the log file to the target device specified in the user exit. After that, the same log file is deleted by the DBMS to save disk space. The Retrieve procedure is called during roll-forward recovery. If the DBMS looks for transactions stored in offline log files, it calls the user exit to retrieve them.

6. Modify the ARCHIVE PROCEDURE as shown in Figure 326.

   Comment out the PKZIP2 line and add the ADSM command line.

```
 if( MODE = '2' & COMPRESSION = 'Y' ) then
 do       /* Add it to LOGS.ZIP in the appropriate database subdirectory

 /*     'pkzip2 -a -es 'ARCH_PATH'\LOGS 'LOG_PATH  */

      'dsmc archive 'LOG_PATH  '-password=mars'
      if rc <> 0 then
        do
       ......
       ........
```

Figure 326. DB2/2 Version 1 ARCHIVE Procedure Modifications

7. Modify the RETRIEVE PROCEDURE as shown in Figure 327 on page 478.

   Comment out the PKZIP2 line and add the ADSM command line.

```
  if COMPRESSION = 'Y' then              /* Compression enable
    do                                   /* Unzip and overwrite the log

/*      'pkunzip2 -o 'ARCH_PATH'\LOGS 'LOG_PATH' 'LOG_FILE  */
'dsmc retrieve 'LOG_PATH||LOG_FILE  '-password=mars'
        ........
        ........
```

Figure 327. DB2/2 Version 1 RETRIEVE Procedure Modifications

8. Save the changes made to INIT, BACKUP, RESTORE, ARCHIVE, and RETRIEVE to SQLUEXIT.CMD.

   The next step is to configure the database you are using to use the user exit program.

   a. Select the Configuration tool icon from the DB2/2 icon box and select the database you want to configure (not shown).

   b. Select Change Database from the Configuration menu. The Change Database Configuration window appears, as shown in Figure 328.

   c. Enable the Log retain and User exit options.

   d. Click on Change to apply these changes.

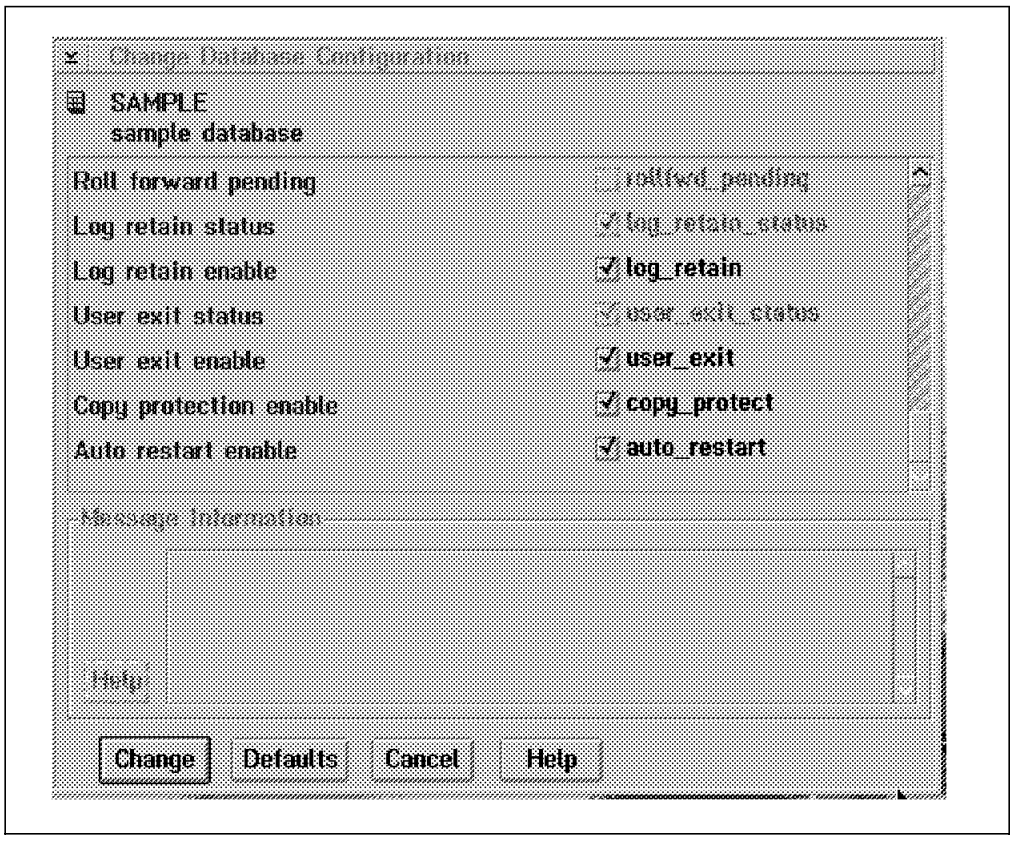


Figure 328. DB2/2 Version 1 Change Database Configuration Window

Everything is now set so that log files are directly moved into ADSM storage as soon as they become inactive.

Please note that when using a user exit to perform your backup, you cannot use user exits that rely on *xcopy*.

14.3.2.2 Backup Example with User Exit Program

Now that you have set up your environment to use the user exit program, you can back up your database. Here are the steps we used in our example to back up the database:

1. Ensure that there is no SQL activity on the database you want to back up. Figure 329 shows the results of the following database command:

   C:> dbm get database status for database sample

   You are ensured there is no SQL activity because the Number of current connects is zero.

```
        Database Status


Normalized time of last backup  = 772200786
Time zone displacement          = -1
Number of current connects      = 0
Database alias                  = SAMPLE
Database name                   = SAMPLE
Database location               = L
Database type                   = DB2/2
Database drive                  = D:
```

Figure 329. Results of DB2/2 Version 1 Database Status Query after a Backup

2. Choose the Recovery Tool icon from the main DB2/2 icon box (not shown).
3. Highlight the database you want to back up and select the Backup function. The Backup Database window appears, as shown in Figure 330 on page 480.
4. Select **0** as the Target Drive, which tells the database manager to use the user exit program.
5. Select Entire Database as the Type of Backup.

   The Changes Only backup type is not allowed when using a user exit program.

6. Click on Backup to start the operation.

   The Message Information box appears and informs you that the operation was successful.

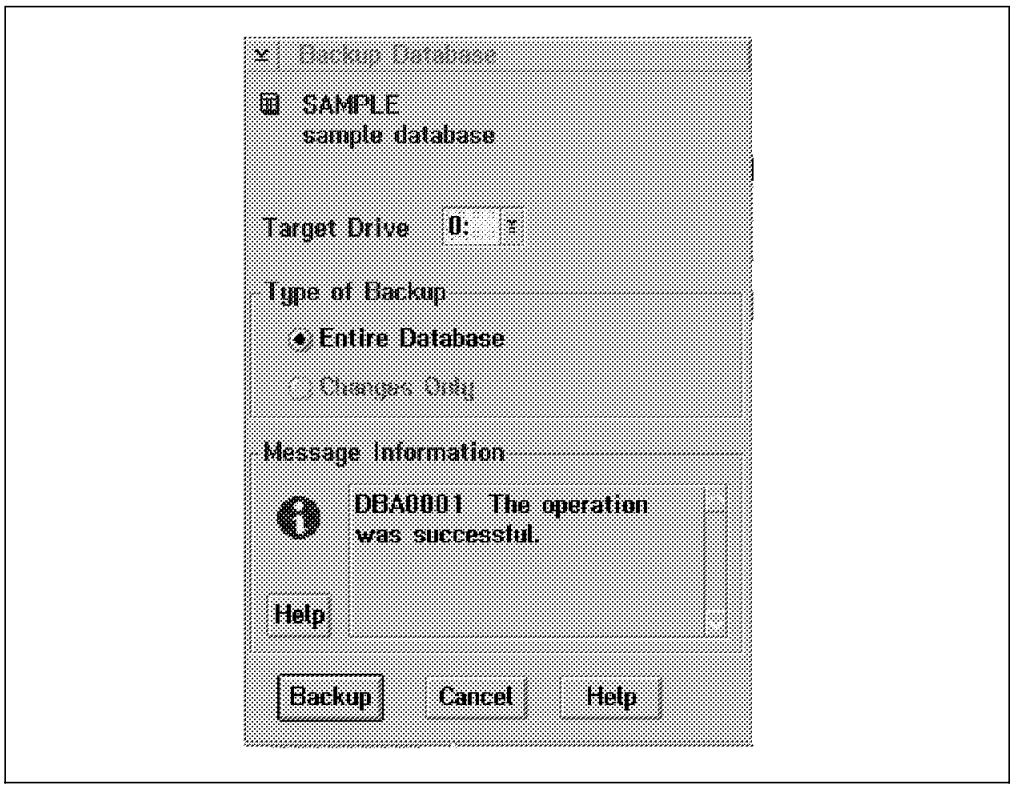


Figure 330. DB2/2 Version 1 Backup Database Window

During the backup process, an OS/2 window shows the PKZIP2 and dsmc command progress. If errors occur, they are logged (as default) in the *C:\USEREXIT* directory, and return codes other than **0** are managed by DB2/2 as errors.

14.3.2.3 Restore Example with User Exit Program

Here are the steps we used in our example to restore a database using the user exit program:

1. Ensure that there is no SQL activity on the database you want to back up. Figure 331 on page 481 shows the results of the following database status command:

   C:>dbm get database status for database sample

   You are ensured there is no SQL activity because the Number of current connects is zero.

```
        Database Status


 Normalized time of last backup  = 772200786
 Time zone displacement          = -1
 Number of current connects      = 0
 Database alias                  = SAMPLE
 Database name                   = SAMPLE
 Database location               = L
 Database type                   = DB2/2
 Database drive                  = D:
```

Figure 331. Results of DB2/2 Version 1 Database Status Query after a Restore

2. Choose the Recovery Tool icon from the main DB2/2 icon box (not shown).
3. Highlight the database you want to recover and select the Recover function. You can decide to overwrite the existing database or create a new one from the recovery operation. As shown in Figure 332 on page 482, we chose to recover an existing database.
4. From the Recover an Existing Database window, select **0** as the Source Drive because you want to use the user exit program.
5. Select the Roll forward option to perform a roll-forward recovery operation. Committed transactions are taken from archived log files and applied to the database according to the strategy you choose. You can specify to read transactions until the end of the online log files or simply until a point in time.

   Needed log files are first retrieved from ADSM and then used by the DBMS.
6. Click on Start to start the process.
7. The Message Information box, as shown, informs you when the operation has successfully completed.

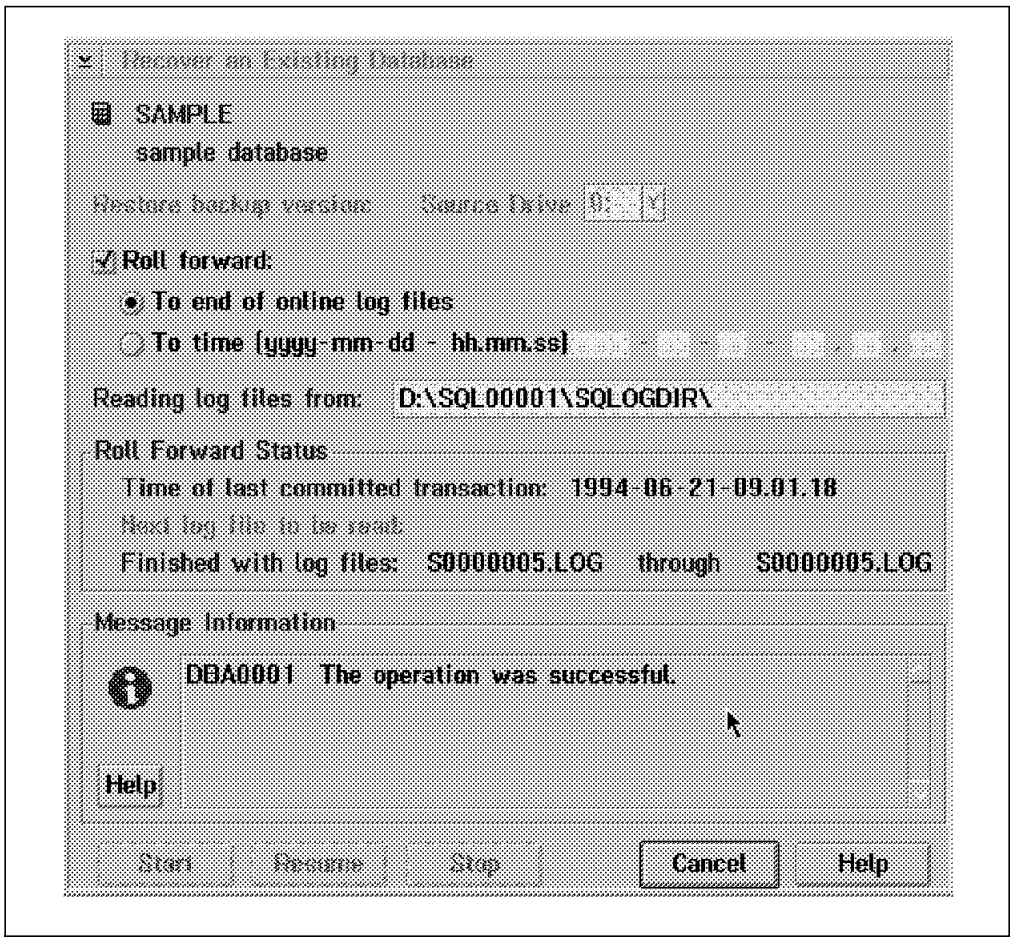


Figure 332. DB2/2 Version 1 Recover an Existing Database Window

During the Restore function, an OS/2 window shows the PKZIP2 and dsmc command progress. If errors occur, they are logged (as default) in the *C:\USEREXIT* directory, and return codes other than **0** are managed by DB2/2 as errors.

14.3.3 Offline Backup of the DB2/2 V1 Database Files Directly with ADSM

Another technique you can use to back up DB2/2 with ADSM is to use ADSM commands directly. This approach saves space because no intermediate files are created, but you must be sure that the database manager is offline before starting the operation in order to preserve data integrity.

Before you can use the ADSM backup command, you need to know the directory where your database has been created. Here are the steps we used in our example:

1. Choose the Directory Tool icon from the DB2/2 icon box (not shown).
2. Select Volume Database from the Directory menu (not shown).
3. Select the drive where you created the database. We choose the **D** drive (not shown).
4. The D: Volume Database Directory window (Figure 333 on page 483) appears with the database name, its location, and the comment associated with it.
5. Write down the directory of your database.

"D:\SQL00001"

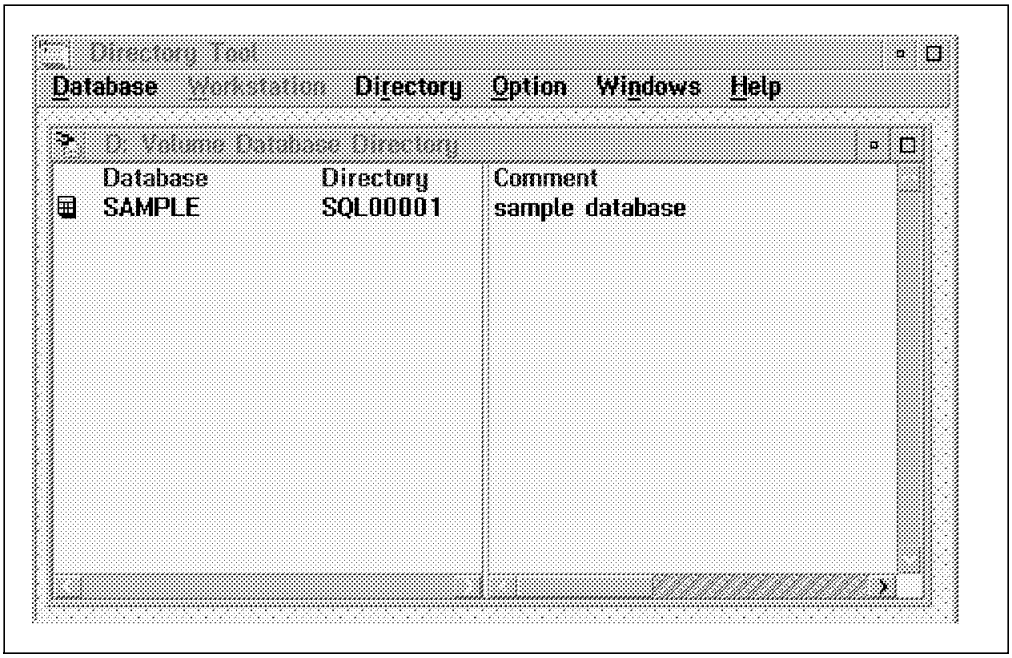


Figure 333. DB2/2 Version 1 D: Volume Database Directory Window

6. Shut down the DBMS using the stopdbm command from a command line.

   C:\ stopdbm

7. Use the ADSM command line or the GUI to back up the database.

   Here is a command line example:

   C:\ dsmc selective -subdir=yes "d:\sql00001*" -password=mars

   The dsmc command will display a list of saved files belonging to the database created in D:\SQL00001.

   To restore an offline backup image, use the dsmc command with the *replace* option set to *YES*. In this case, the command overwrites the existing files. We used this option to save space because we have many saved files.

   C:\ dsmc restore -replace=yes -subdir=yes "d:\sql00001*" -password=mars

14.4 Using ADSM to Back Up DB2/2 Version 2.1.1 +

Starting with DB2/2 Version 2.1.1, ADSM is supported in the same way as DB2/6000 Version 2. DB2/2 V2.1.1 uses the ADSM API to send data directly to ADSM; full or tablespace, offline or online backups are supported. We show an example of an offline backup to ADSM and then a restore of that backup. We use both the command line and the new *Database Director*.

At the time of writing, we tested with a beta version of DB2/2 V2.1.1. DB2/2 supports only the ADSM password generate function, not password prompt, if you use authentication. Password generate support is provided in PTF2 of the ADSM OS/2 API and was not available when we wrote this section. Therefore, in our example we turned off password authentication. If you do not turn password authentication off, your backup will fail with return code 137. The *open x*

sessions parameter is not available because of an incompatibility between DB2/6000 Version 2 and the ADSM Client. When a thread-safe version of the ADSM Client is available, the *open x sessions* parameter will also be available.

14.4.1 Setting Up an ADSM OS/2 Client and DB2/2 Version 2

The installation and setup of the ADSM OS/2 client and DB2/2 Version 2.1.1 is straightforward. You must ensure that the API environment variables set by ADSM are correct, in particular the location of your options file. Figure 334 shows the API environment variables.

```
DSMI_CONFIG=C:\ADSM\API\DSM.OPT
DSMI_DIR=C:\ADSM
DSMI_LOG=C:\ADSM
```

Figure 334. ADSM API Environment Variables for DB2/2 Version 2

One indication that your DSMI_CONFIG variable is incorrect is if your backup fails with return code 2220; This return code means that ADSM cannot find the DSM.OPT file. Please note that ADSM installation sets the DSMI_CONFIG environment variable to C:\ADSM\API\DSM.OPT whereas the default location for DSM.OPT is C:\ADSM. For our redbook configuration we copied the DSM.OPT file into the API directory, but for consistency, the environment variable should be corrected and the OS/2 machine re-booted.

14.4.2 Offline Backup of DB2/2 Version 2 Using the Command Line

This example shows an offline backup using the DB2/2 Version 2 command line processor (CLP).

Log on to the OS/2 workstation as an administrator and check that no applications are using the database that you want to back up. Use the command in Figure 335 to ensure that the database is not in use. If the database is not in use, you should see the message as shown in Figure 335.

```
db2 list applications for database sample
SQL1611W  No data was returned by Database Systems Monitor.
```

Figure 335. Ensure the DB2/2 Version 2 Database Is Not in Use

If any applications are listed, you must terminate them, or force them using the following command:

```
db2 force applications all
```

Start the backup operation from the command line:

```
db2 backup database sample use adsm
```

The *use adsm* option tells DB2/2 Version 2 to use the ADSM API and therefore sends the output backup file to ADSM instead of using common devices.

The backup is complete when the message shown in Figure 336 on page 485 appears.

```
Backup successful. The timestamp for this backup image is : 19951101095358
```

Figure 336. DB2/2 Version 2 Backup Completion Message

You can use the DB2/2 Version 2 *list history* command to display the time stamps for any backups that you have taken (see Figure 337).

```
                    List Recovery History File for HISTORY

Number of matching file entries = 1


 Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log  Backup ID
+-- --- ------------------ ---- --- ------------ ------------ -------------+
| B  D  19951101095358001   F    A  S0000000.LOG S0000000.LOG                |
+---------------------------------------------------------------------------+
| Contains 2     tablespace(s):                                             |
|                                                                           |
| 00001 SYSCATSPACE                                                         |
| 00002 USERSPACE1                                                          |
+---------------------------------------------------------------------------+
| Comment: DB2                                                              |
+---------------------------------------------------------------------------+
  00001 Location: dll\db2adsm.dll
```

Figure 337. DB2/2 Version 2 List History Command Results

This display also shows other useful information such as the earliest and current logs used for this database. This log information is used if your database is enabled for roll-forward recovery.

14.4.3 Offline DB2/2 V2.1.1 Recovery Using the Command Line

To restore a database using the DB2/2 Version 2 CLP you must perform the following steps:

1. Log on to the OS/2 workstation as an administrator.
2. Obtain the time stamp of the database backup from which you want to restore from. You can obtain the time stamp by using the *list history* command.
3. Start the recovery operation from the command line:

```
db2 restore database sample use adsm taken at 19951101095358
```

If you are restoring the backup file to a database that already exists, you will be asked to confirm that the existing database can be overwritten, as shown in Figure 338 on page 486. If a time stamp is not provided, the most recent backup image is used by default. The *open x sessions* parameter is not available because of an incompatibility between DB2/6000 Version 2 and the ADSM Client. When a thread-safe version of the ADSM Client is available, the *open x sessions* parameter will also be available.

```
  SQL2539W  Warning!  Restoring to an existing database that is the same
  as the backup image database.  The database files will be deleted.
  Do you want to continue (y/n)  y

DBA0000I  The RESTORE DATABASE command completed successfully.
```

Figure 338. DB2/2 Version 2 Command Line Restore

14.4.4 Offline DB2/2 V2.1.1 Backup Using the Database Director

This example shows an offline backup using the Database Director. Start the Database Director by double-clicking on the Database Director icon or entering *db2dd* on the command line. The window shown in Figure 339 is displayed. This window is shown after steps 1-3 are completed.

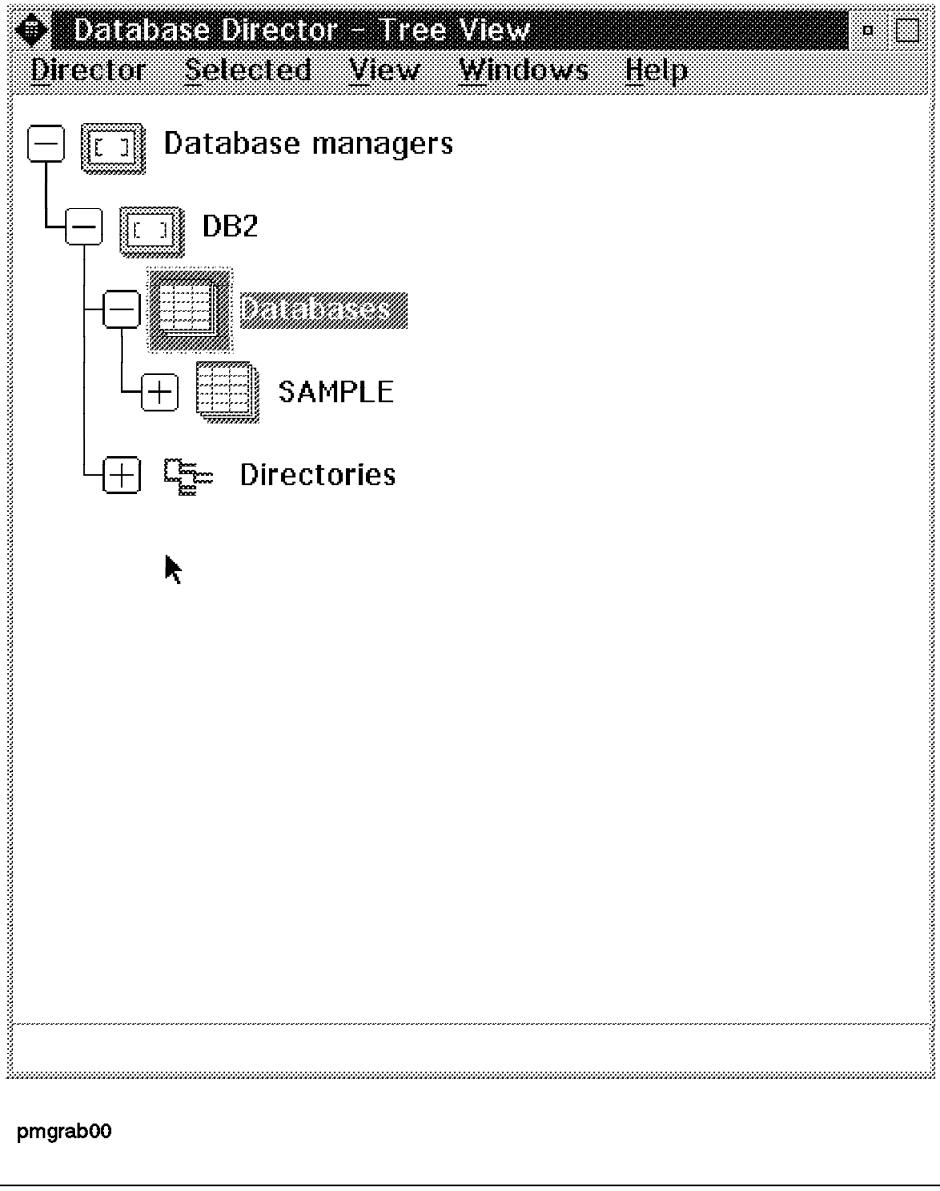


Figure 339. DB2/2 Version 2 Database Director

To start a database backup:

1. Click on the plus sign next to DB2 (not shown). A minus sign appears next to DB2.
2. Click on the plus sign next to Databases (not shown). A minus sign appears next to DB2.
3. Click on the database you want to back up.
4. Select *Selected* from the action bar and click on *Back Up...* in the pull-down menu.

The backup window shown in Figure 340 appears.

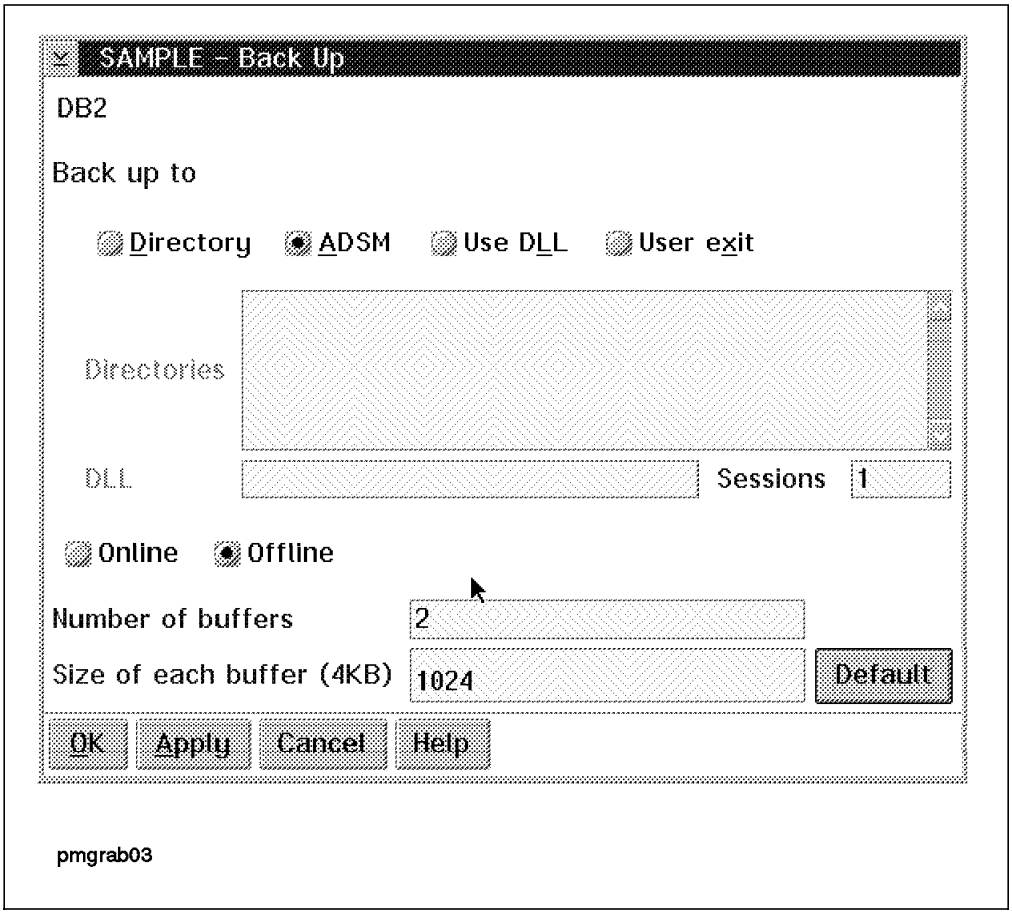


Figure 340. DB2/2 Version 2 Backup Menu

You do not have to change any options on this window to perform a backup. The default buffer size uses the *backbufsz* configuration value. You can display this value by clicking on the *Default* button.

Click on OK to start the backup. An information window appears (Figure 341 on page 488) and informs you of the job number. Click on OK to remove the information window.

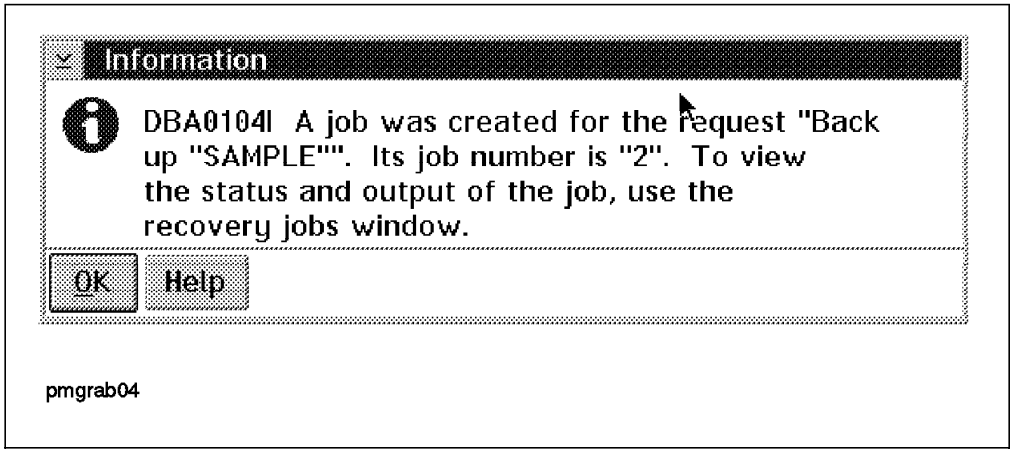


Figure 341. DB2/2 Version 2 Information Window

The Recovery Utility is a separate graphical program used to monitor jobs submitted by the Database Director. To start the Recovery Utility click on the appropriate icon in the DB2/2 folder. Figure 342 shows the *Jobs - Details View* display, which includes the database backup job submitted in Figure 340 on page 487 and Figure 341.

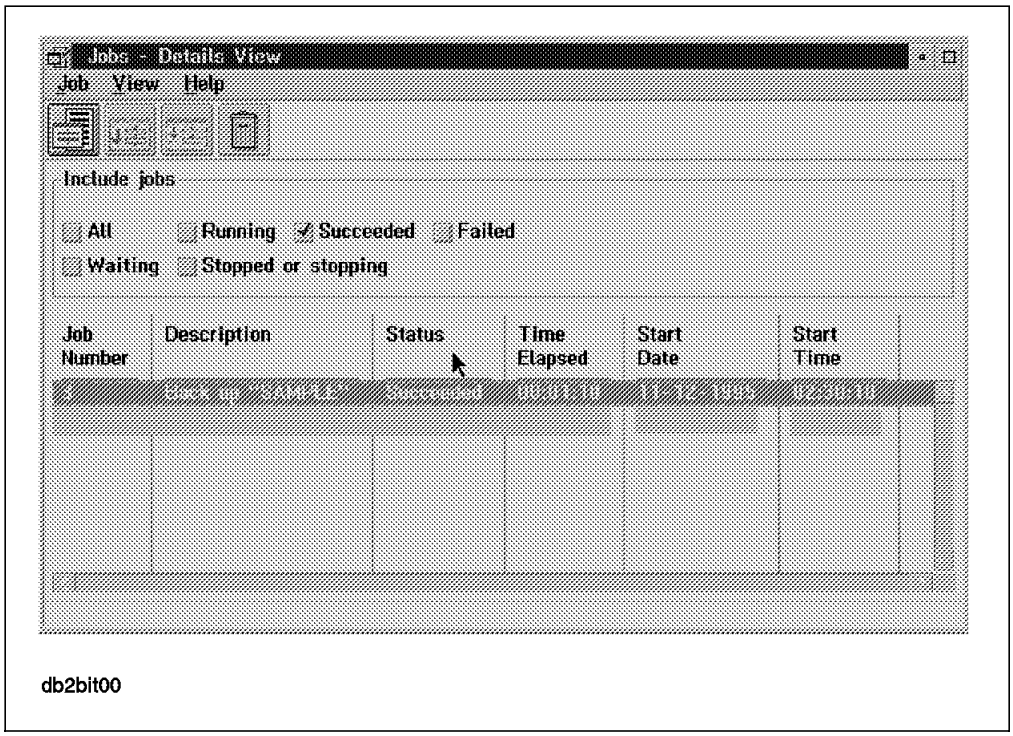


Figure 342. DB2/2 Version 2 Database Backup Jobs

When the backup job is completed you can double-click on the job line to see the job output as well as the time stamp of the backup (Figure 343 on page 489).

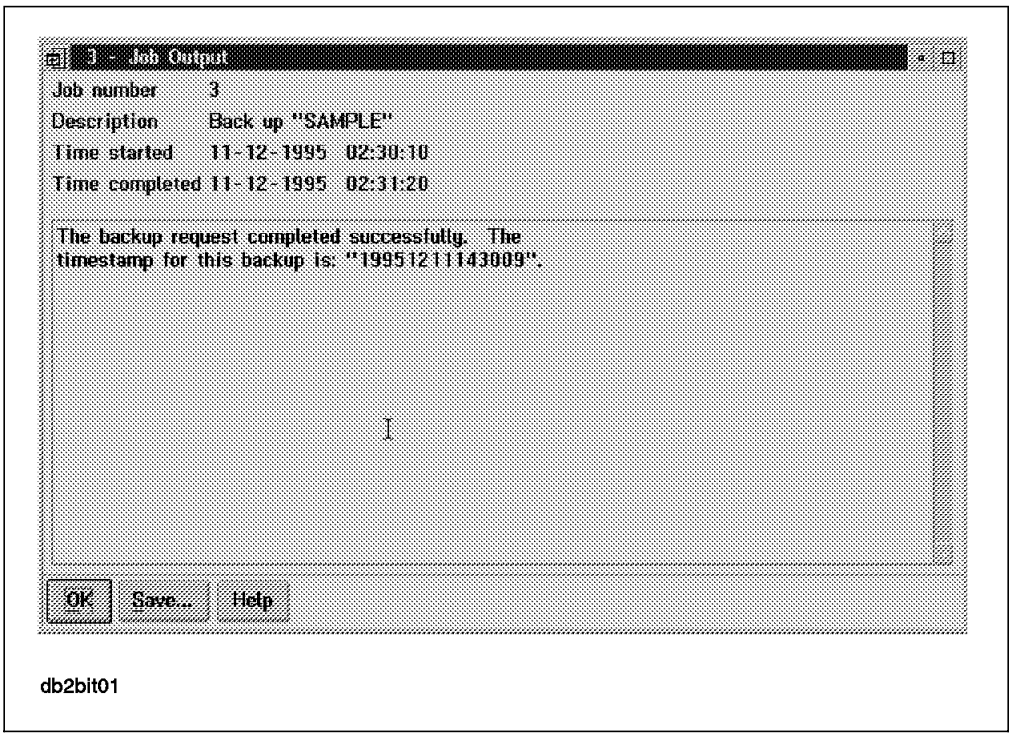


Figure 343. DB2/2 Version 2 Backup Job Output

14.4.5 Database Recovery with Database Director

To recover a database from ADSM with the Database Director, start the Database Director as previously described. Select the database that you want to recover, select *Selected* from the action bar, and click on *Recover...* from the pull-down menu. This action will display the Recover Database notebook shown in Figure 344 on page 490.

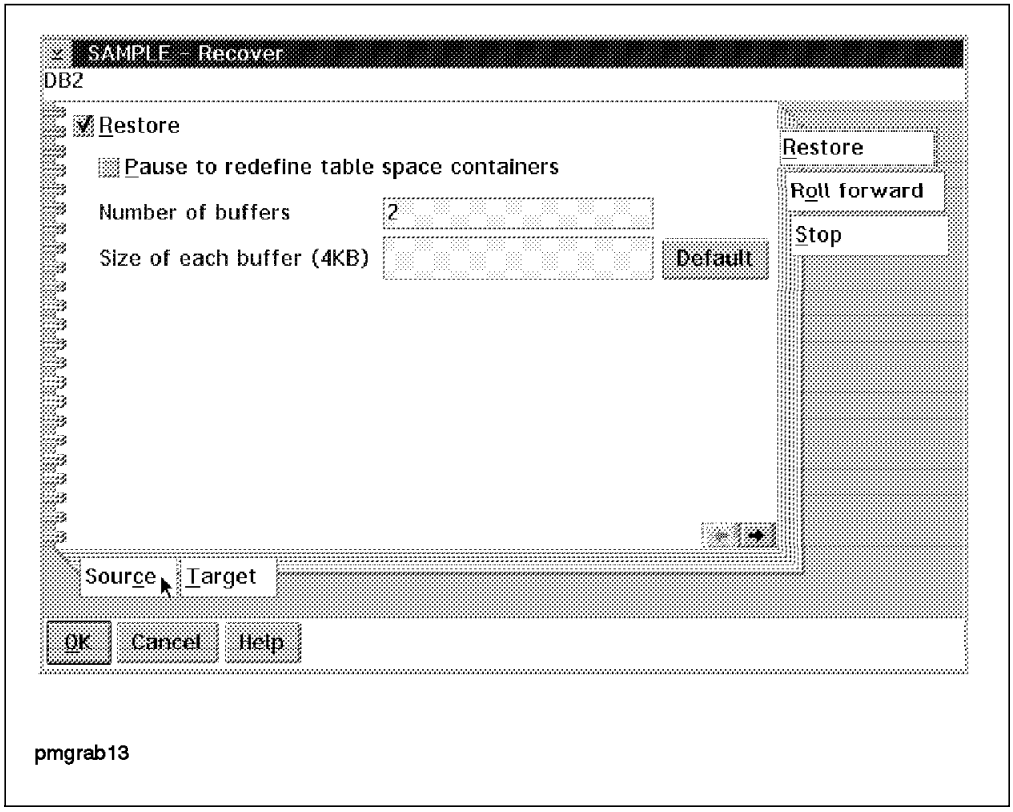


Figure 344. DB2/2 Version 2 Recover Database Notebook

Click on the *Source* notebook tag and enter the date and time of your backup as shown in Figure 345 on page 491. This is the date and time as recorded in the time stamp. For example, if the time stamp is 19951101095358, the date of the backup is 11-01-1995, and the time of the backup is 09:53:58.

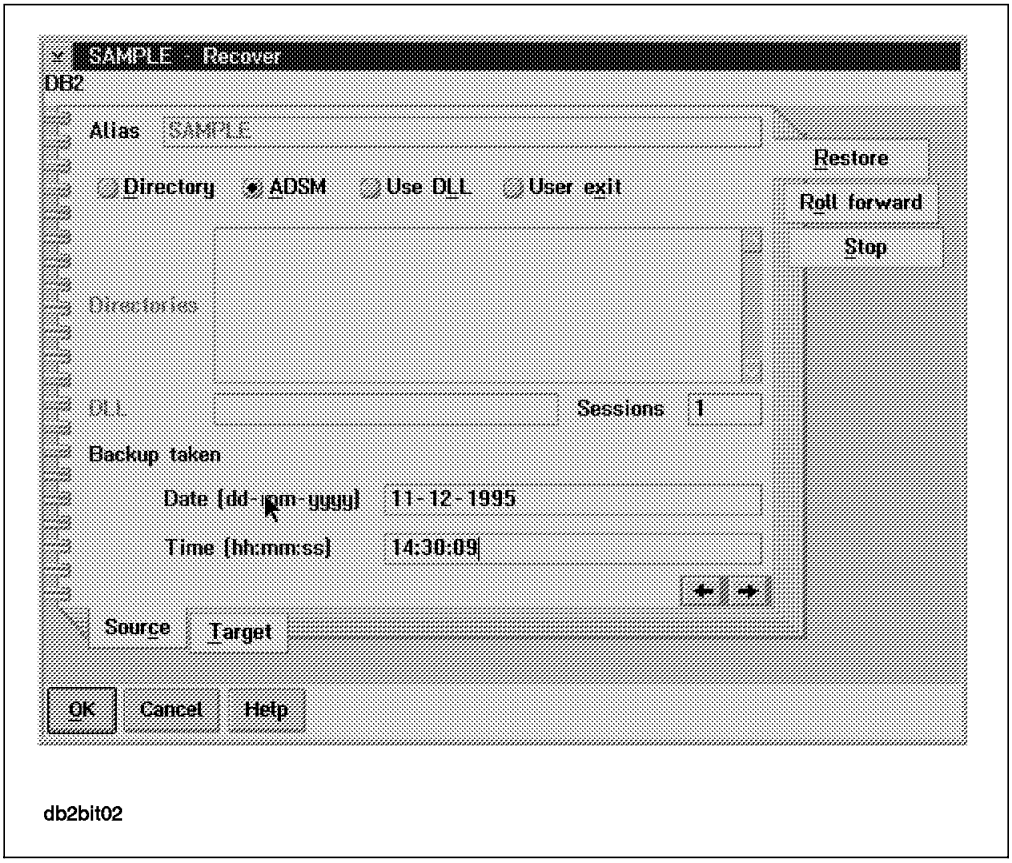


Figure 345. DB2/2 Version 2 Recover Source Notebook Page

Click on OK to start the recover process. An information window appears, as shown in Figure 346, to inform you of the recovery job number.

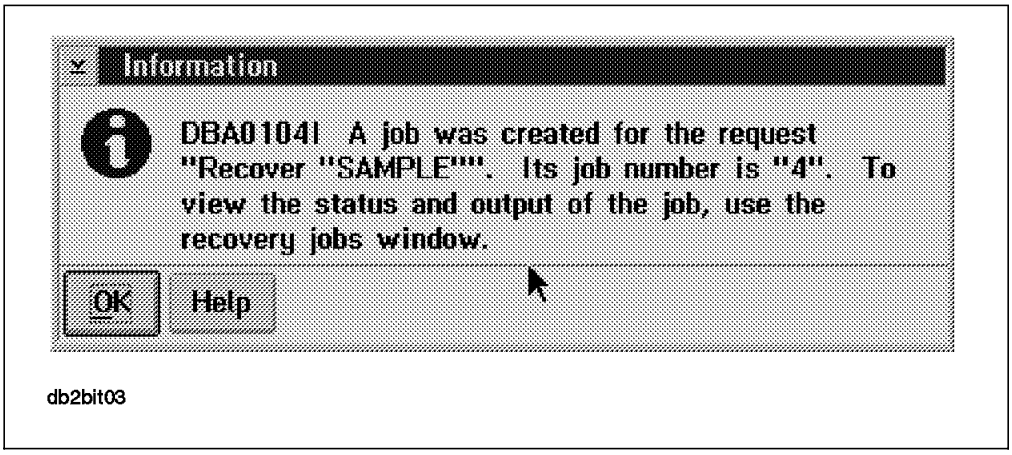


Figure 346. DB2/2 Version 2 Information Window

Use the recovery utility to follow the progress of the job. The recovery utility is a separate graphical program used to monitor jobs submitted by the Database Director. To start the recovery utility click on the appropriate icon in the DB2/2 folder. Figure 347 on page 492 shows the *Job - Details View* display, which includes the database restore job we submitted in Figure 345.

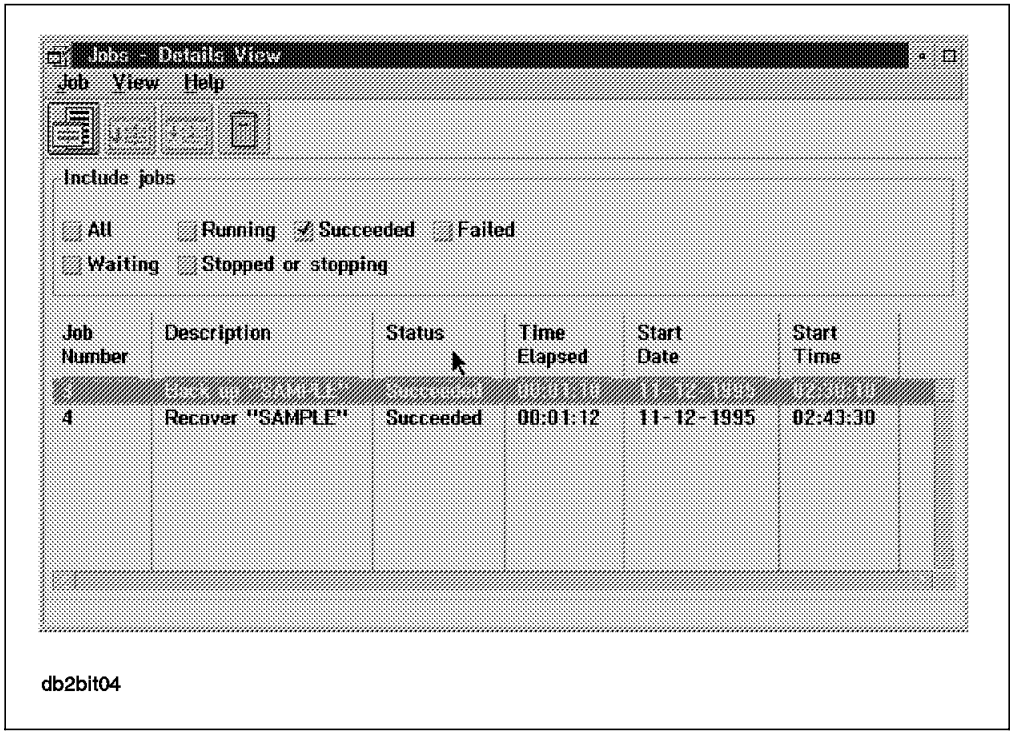


Figure 347. DB2/2 Version 2 Restore Job Details

When the restore job is completed you can double-click on the job line to see the job output (Figure 348).

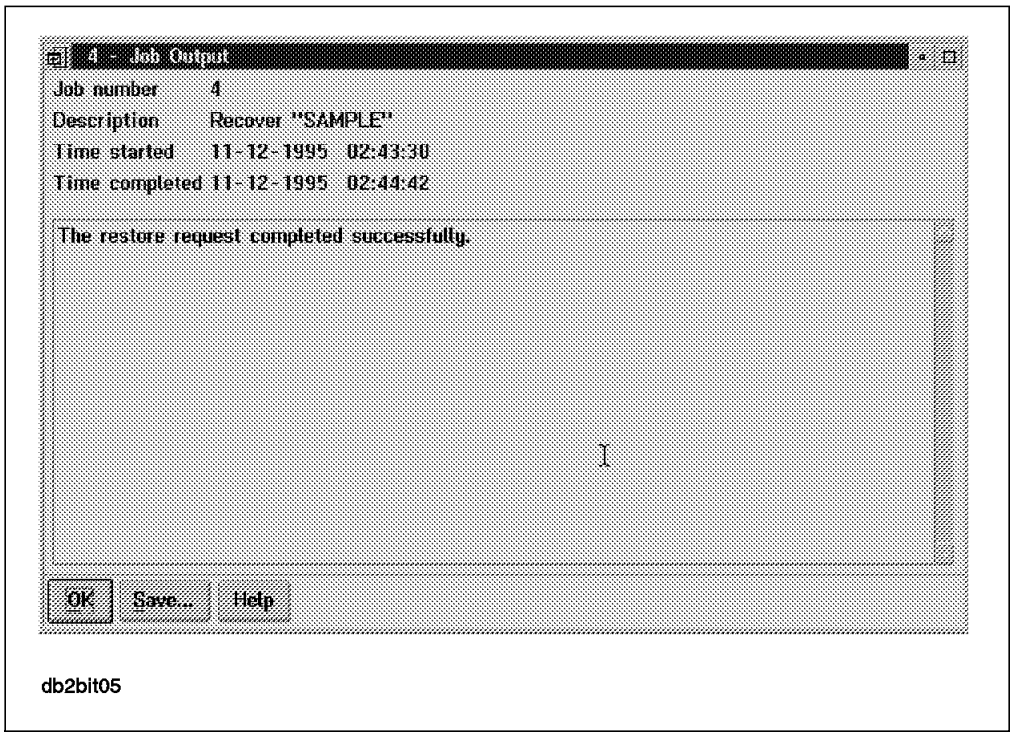


Figure 348. DB2/2 Version 2 Restore Job Output

14.5 Additional Considerations

Let us briefly look at the *quiesce* option for backups with DB2/2 Version 1.2 and how to query your ADSM server.

14.5.1 Using the Quiesce Option for Version 1.2 Backups

As discussed in 14.2, “DB2/2 Backup Utilities” on page 466, DB2/2 Version 1.2 provides a significant enhancement to the Version 1.1 backup utility. The DBA can perform a backup of a database with active connections, using a new quiesce option. This means that users, in addition to the DBA, remain connected to the database. You can think of this as somewhere in between an offline and online backup. Quiesce is supported either through the DB2/2 API or command line interface.

Here is an example using the command line interface:

To back up our SAMPLE database without forcing users off, use the DB2/2 backup utility with the following parameters:

```
C:> backup database SAMPLE to A WITH QUIESCE ERROR NEW TRANSACTIONS
```

This command performs an offline backup of the active SAMPLE database. The ERROR NEW TRANSACTIONS option, results in users receiving an error as soon as they start new SQL operations.

14.5.2 Querying Facility for the ADSM Server

To query the ADSM server, you can use either the command line, the ADSM GUI, or the *db2adutl* utility, which is documented in section 13.7.2, “Managing ADSM Objects Created with DB2/6000 Version 2” on page 457.

Note

If you use the ADSM backup/archive client command line or GUI, you will see only those files that were backed up or archived by the ADSM backup/archive client. Similarly, the db2adutl utility shows only those files that were backed up in DB2/2 Version 2 through DB2 backup utilities and the ADSM API.

14.5.2.1 Command Line

Here is a sample command used to query the data stored in the ADSM storage pools:

```
C:> dsmc query backup -Subdir=yes d:\db_backs\* -password=mars
```

Figure 349 on page 494 shows the results of the command.

```
Version 1.x
 Command Line Backup Client Interface - Version 1, Release 1, Level 0
 5648-020 (C) Copyright IBM Corporation, 1990, 1993, All Rights Reserved.

 Session established with server BALTIC: AIX-RS/6000
  Server Version 1, Release 2, Level 0.0
  Server date/time: 06/17/1994 15:44:29  Last access: 06/17/1994 15:44:00

    Size      Backup Date     Mgmt Class A/I File
    ----      -----------     ---------- --- ----
  83,881  06/08/1994 16:32:54   DEFAULT  A  D:\DB_BACKS\SAMPLE1\SAMPLE.ZIP
 102,448  06/11/1994 16:21:46   DEFAULT  A  D:\DB_BACKS\SAMPLE2\SAMPLE1.ZIP
  98,720  06/11/1994 08:55:41   DEFAULT  A  D:\DB_BACKS\SAMPLE\SAMPLE2.ZIP
```

Figure 349. Results of ADSM Query Command for Storage Pools for DB2/2 Version 1

Here is the command to query the log files that were archived in ADSM server storage:

```
C:> dsmc query archive -Subdir=yes "d:\sql00003" -password=mars
```

Figure 350 shows the results of the command.

```
 Command Line Backup Client Interface - Version 1, Release 1, Level 0
 5648-020 (C) Copyright IBM Corporation, 1990, 1993, All Rights Reserved.

 Session established with server BALTIC: AIX-RS/6000
  Server Version 1, Release 2, Level 0.0
  Server date/time: 06/17/1994 15:44:29  Last access: 06/17/1994 15:44:00
Size   Archive Date - Time   File - Expires on - Description
----   -------------------   ------------------------------
13,312 06/11/1994 16:23:50   D:\SQL00003\SQLOGDIR\S0000000.LOG06/11/95
17,408 06/11/1994 16:24:12   D:\SQL00003\SQLOGDIR\S0000001.LOG06/11/95
```

Figure 350. Results of ADSM Query Command for DB2/2 Version 1 Log Files

14.5.2.2 Graphical User Interface

Here is an example of how to query the ADSM server using the GUI of the backup/archive client.

From your ADSM client select Restore by directory tree to see your backed up files. You can also see both active and inactive backup files so that you can decide which version of the backup image to use for a restore. Figure 351 on page 495 shows the backed up files.

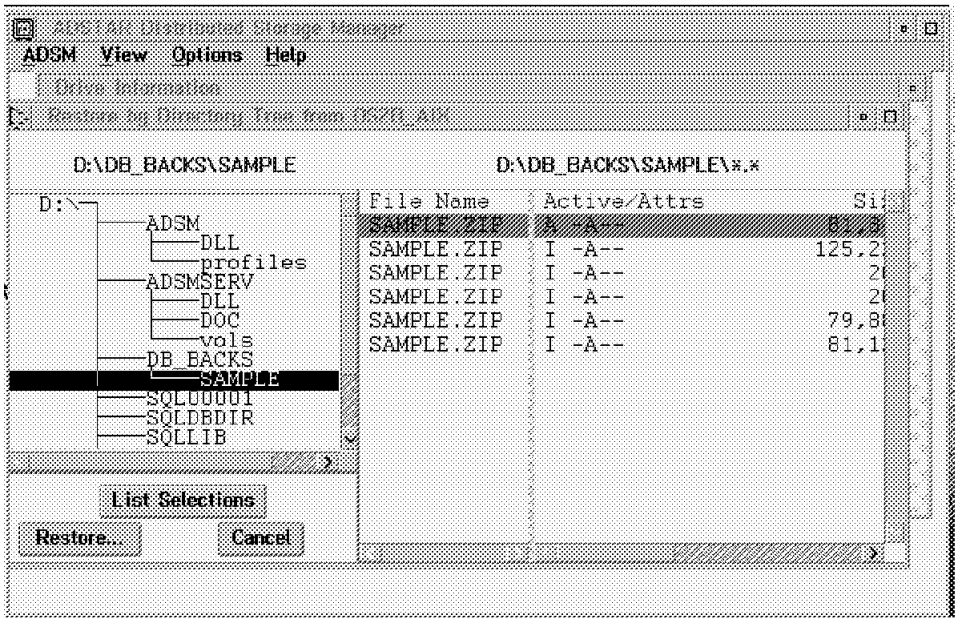


Figure 351. DB2/2 Restore by Directory Tree Window

To see the log files you archived, select the Retrieve option from the main ADSM backup/archive client window. Figure 352 shows the results.

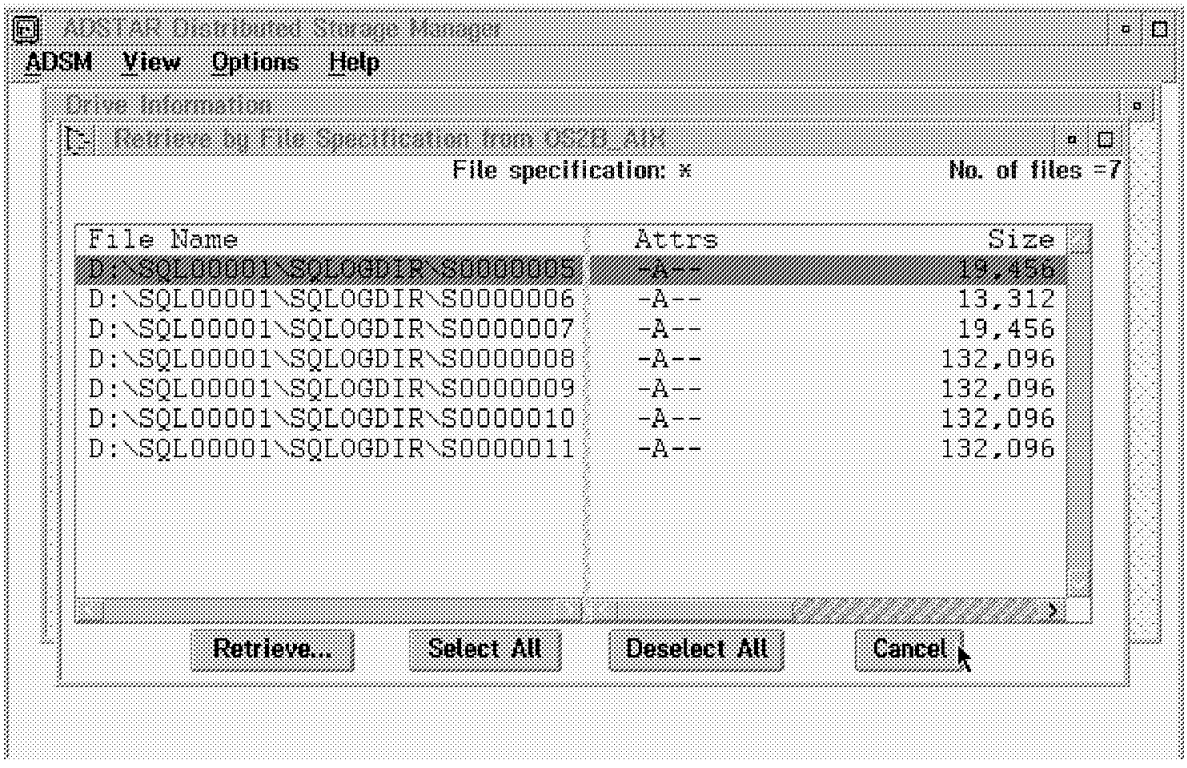


Figure 352. Retrieve by File Specification Window

Chapter 15. ADSM and DataHub

This chapter describes DataHub, an IBM database management product available for the UNIX (DataHub for UNIX OS) and OS/2 (DataHub for OS/2) platforms. We give an overview of the DataHub features and explain how you can use DataHub to back up databases through ADSM. We then look at how to set up DataHub for UNIX OS and DataHub for OS/2. Finally, we give examples of backing up DB2/6000, Sybase, and Oracle databases with DataHub for UNIX OS and DB2/6000 and DB2/2 databases with DataHub for OS/2.

15.1 What Is DataHub?

DataHub is designed to help you manage multiple databases across a number of platforms. With DataHub you can manage relational databases distributed across a network. You can manage the hosts on which the databases reside, and you can manage DataHub itself.

DataHub has a central machine known as the *control point* in the UNIX version or the *DataHub workstation* in the OS/2 version, as shown in Figure 353 on page 498. This central machine consists of the DataHub software and a database that contains information about the objects that DataHub manages.

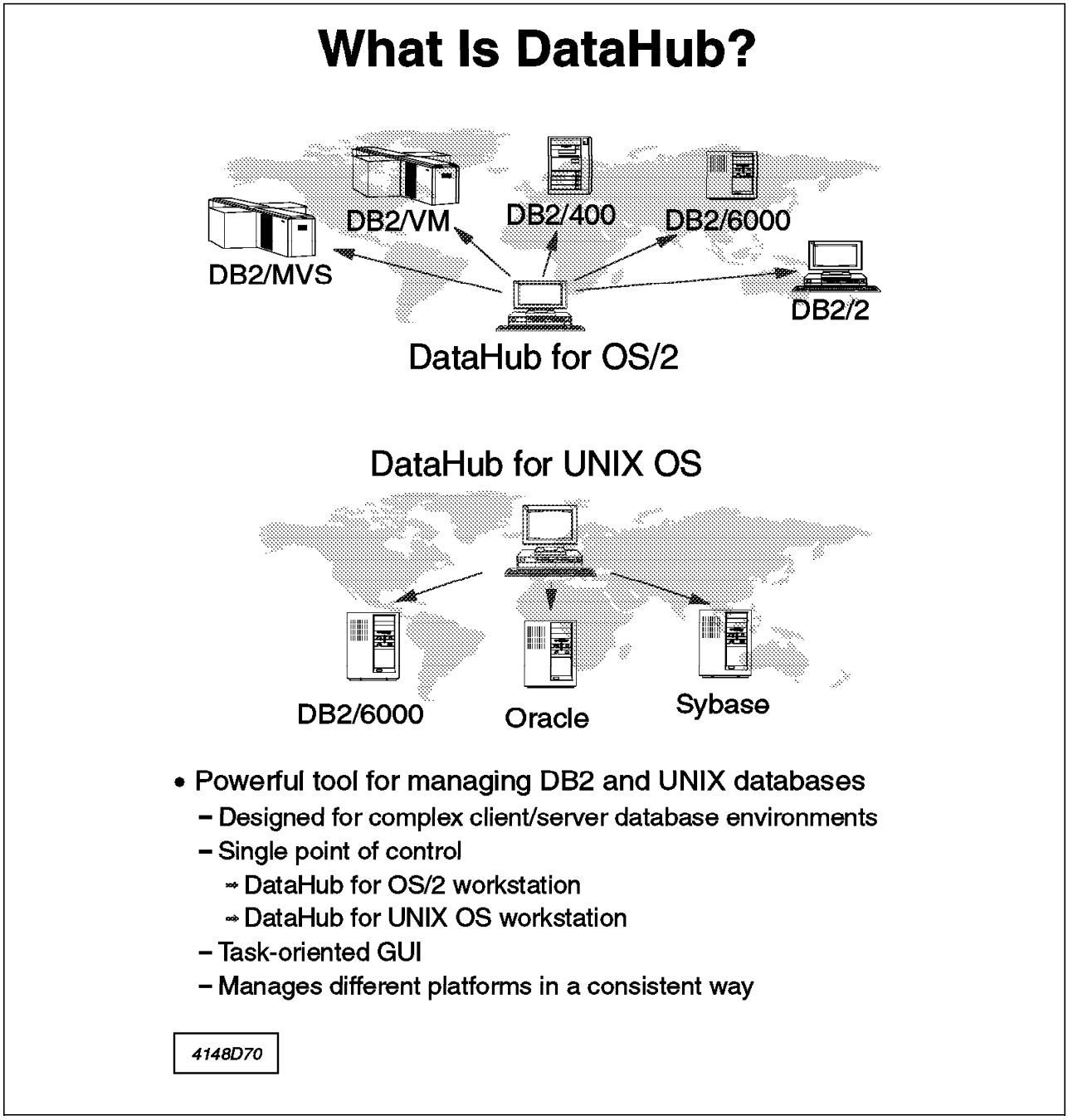


Figure 353. DataHub Environment

The remote machines that DataHub for UNIX OS manages are known as *managed hosts*. Each machine can contain one or more databases that DataHub for UNIX OS manages. The managed hosts run programs called *watchdogs*, which monitor functions of the managed host and the database. The watchdogs are configurable to monitor almost any event.

Figure 354 on page 499 shows the DataHub for UNIX OS components, including the managed host and watchdogs.

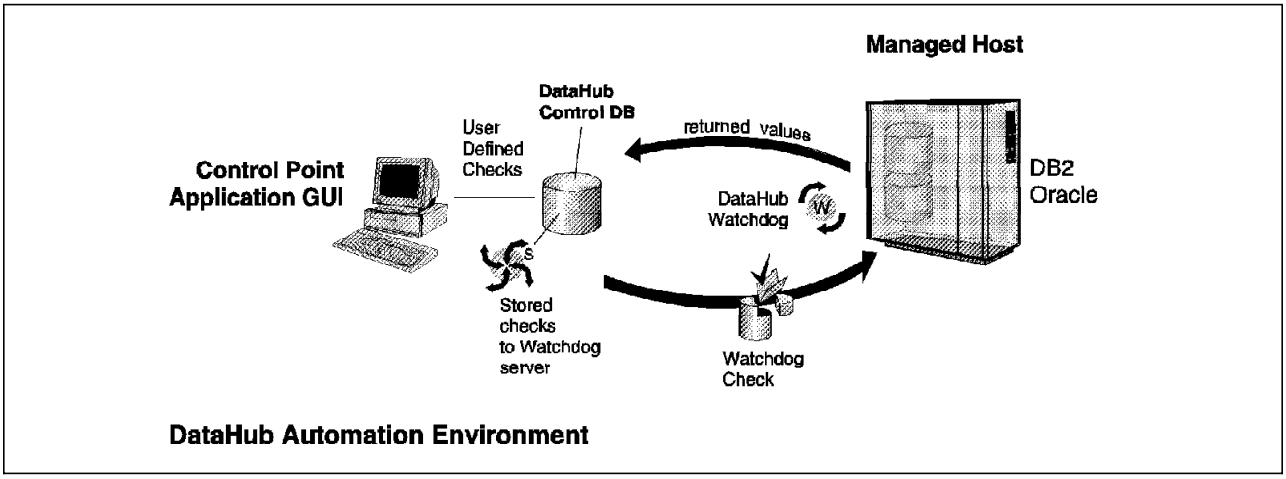


Figure 354. DataHub for UNIX OS Components

15.2 DataHub ITSO Test Environment

We set up DataHub for UNIX OS under AIX with DB2/6000, Sybase, and Oracle managed databases. The DataHub control point database was set up using DB2/6000 Version 2.1. We set up DataHub for OS/2 under OS/2 Warp V3.0 with DB2/6000 and DB2/2 Version 2 managed databases. We set up the DataHub workstation database, using DB2/2 Version 2.1.

15.2.1 DataHub for UNIX OS Installation Notes

Installation of DataHub for UNIX OS is described in the *DataHub for UNIX OS Installation and Configuration Guide*. As part of the installation you are prompted for all of the information necessary to set up the DataHub control point database and the UNIX userid and password.

To start DataHub for UNIX OS you must log on to your UNIX system with the DataHub userid and enter *datahub*. The DataHub for UNIX OS Operating Systems window is displayed, as shown in Figure 355 on page 500.

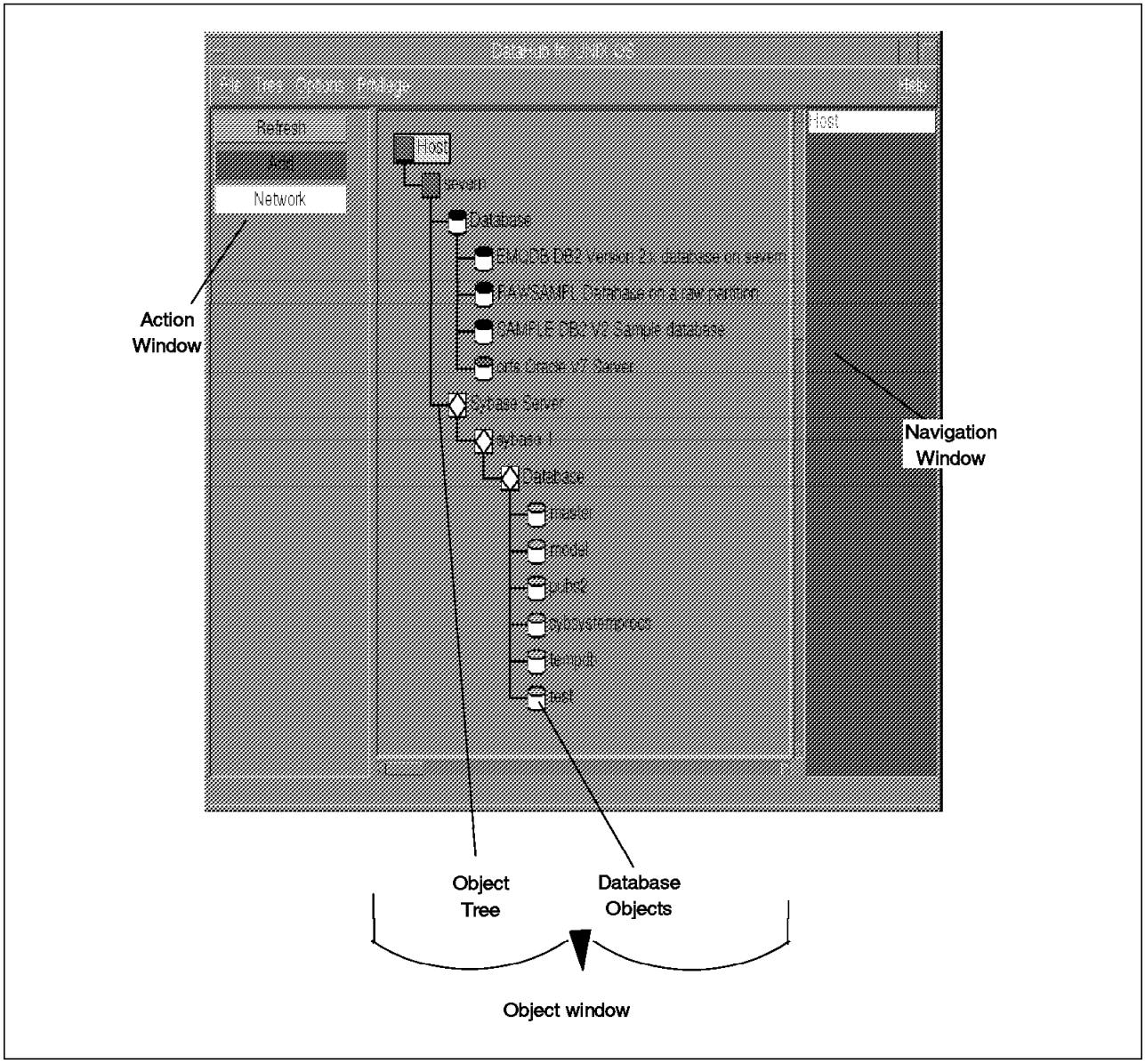


Figure 355. DataHub for UNIX OS Operating Systems Window

The DataHub for UNIX OS Operating Systems window has three sections:

- Object window

  The *Object window* shows a hierarchical view of the host machines, the database managers installed on those machines, and the databases that the database managers manage. The view is called the *object tree*. You use the object tree to select the item that you want to manage and expand the tree to show its contents.

- Action window

  When you select an object from the object tree, the actions that you can perform on that object are shown in the *Action Window*. The actions vary according to the database product and the specific functions available for each database.

- Navigation window

The *Navigation window* shows the path of objects between the host and the selected object on that host. This window is useful when the object tree grows and the host may no longer be visible on the object tree view. You can use the Navigation window to quickly “climb” back up the object tree.

15.2.2 DataHub for OS/2 Installation Notes

Installation of DataHub for OS/2 is relatively simple. We encountered some problems, however, when installing DataHub for OS/2 AIX support. To install the AIX support to manage a remote DB2/6000 database you must perform the following steps:

1. Install the DataHub Support code using SMIT.
2. Ensure that DB2/6000 is installed and configured on the AIX machine.
3. Create a link to the libdb2.a library from /lib to enable the DataHub setup code to work:

   ```
   cd /lib
   ln -sf /usr/lpp/db2_02_01/lib/libdb2.a
   ```

   You must be logged on as root.
4. Run the DataHub setup program to create a DataHub configuration file:

   ```
   cd /usr/lpp/datahub
   ./emqsetup.sh
   ```

   Run the setup program as root.

   Make sure that you read the prompts carefully as the values you enter must be correct for DataHub to function.
5. Change ownership of the DataHub log directory to give your DB2/6000 administrator userid authority to write DataHub error logs:

   ```
   cd /usr/lpp/datahub
   chown db26000.sysadm probdet
   ```
6. Log on to your machine as the DB2/6000 administrator (in our case, db26000). You must use the DB2/6000 administrator userid to perform the remaining steps.
7. Catalog your DataHub workstation and the DataHub database on your AIX machine. For example:

   ```
   db2 catalog tcpip node puget REMOTE puget server db2os2c
   db2 catalog database emqdb at node puget authentication server
   ```
8. Bind the DataHub support to the database (you only have to do the bind this once, no matter how many AIX machines you are managing):

   ```
   cd /usr/lpp/datahub/platform
   db2 connect to emqdb
   db2 bind emq6ktml.bnd grant public blocking all
   db2 bind emq6ktsq.bnd grant public blocking all
   db2 bind emq6cf21.bnd grant public blocking all
   ```
9. Run the emqconf program to add information to the DataHub database and create the dhhost name on the DataHub OS/2 Workstation. When this has completed successfully, check the AIX host name on the OS/2 workstation as shown in Figure 356 on page 502. We found that emqconf set the TCPIP address to emq6t, which is not the correct address If emqconf sets the TCPIP address incorrectly, delete the entry and add the correct entry manually.

```
[C:]dhhost list bering
Hostname =  BERING   operating system type =  AIX
 Configured for TCPIP address =  emq6t

Returning from DHHOST, rc = 0

[C:\]dhhost delete bering
Returning from DHHOST, rc = 0

[C:\]dhhost add bering aix tcpip 9.113.36.145
Returning from DHHOST, rc = 0

[C:\]dhhost list bering
Hostname =  BERING   operating system type =  AIX
 Configured for TCPIP address =  9.113.36.145

Returning from DHHOST, rc = 0
```

Figure 356. Correcting the Managed Host TCPIP Address for DataHub for OS/2

You must restart DataHub to pick up the new host.

15.2.3 DataHub for OS/2 Main Window

Figure 357 shows the DataHub for OS/2 workstation main window.

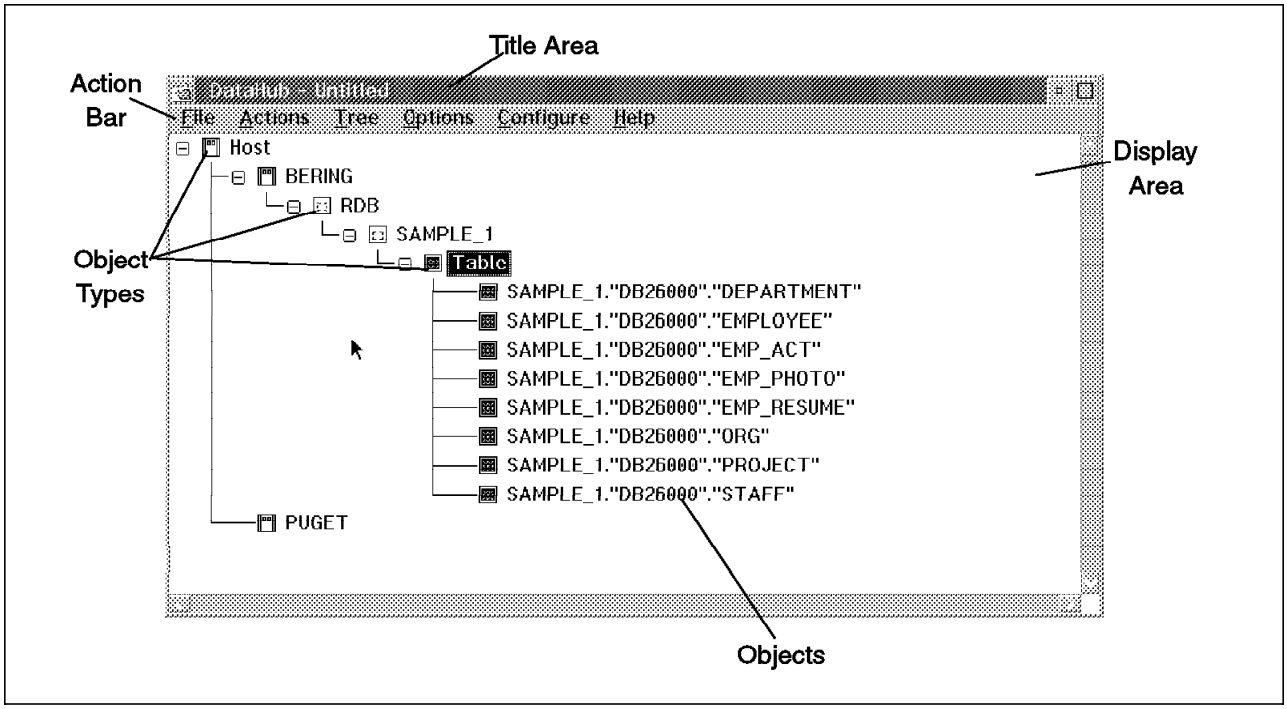


Figure 357. DataHub for OS/2 Operating Systems Window

The DataHub for OS/2 main window follows the standard format for OS/2 Workplace Shell windows in Figure 357.

You have the option of saving your window, once it has been created, so that you can reload the display at any time. We recommend that you save your DataHub display with only the host objects displayed so that you have a

convenient starting point for displaying databases. DataHub sets up sessions to some databases (DB2/2 in particular) to obtain information about objects; the sessions that DataHub sets up could prevent you from performing certain actions that require the database to be offline. See 15.11, “Backing up DB2/2 with DataHub for OS/2” on page 522 for an example. Saving a copy of the display and reloading it when necessary is one way to clear these sessions.

The object tree in the display area consists of *object types* and *objects*.

15.3 Backing Up DB2/6000 with DataHub for UNIX OS

To back up a DB2/6000 database using DataHub for UNIX OS, perform the following steps (for this example we are using DB2/6000 Version 2):

1. When the DataHub for UNIX OS window appears, select the host you want to manage and click on *Display* in the Action window (not shown).
2. Select Database from the *Select One or More* window and click on OK (not shown).
3. Select the database you want to back up. In our example we select the SAMPLE database (not shown).
4. Click on the Backup button in the Action window (not shown).

   The *Operating System Login for Database Operation* window appears. This window is used to provide verification for certain database, database manager, and operating system procedures. Every time a database (management) command is executed, a connection is set up. The *User* and *Ppassword* combination is necessary for this connection. Enter the userid and password and click on OK.

   The Backup Database window appears as shown in Figure 358 on page 504.

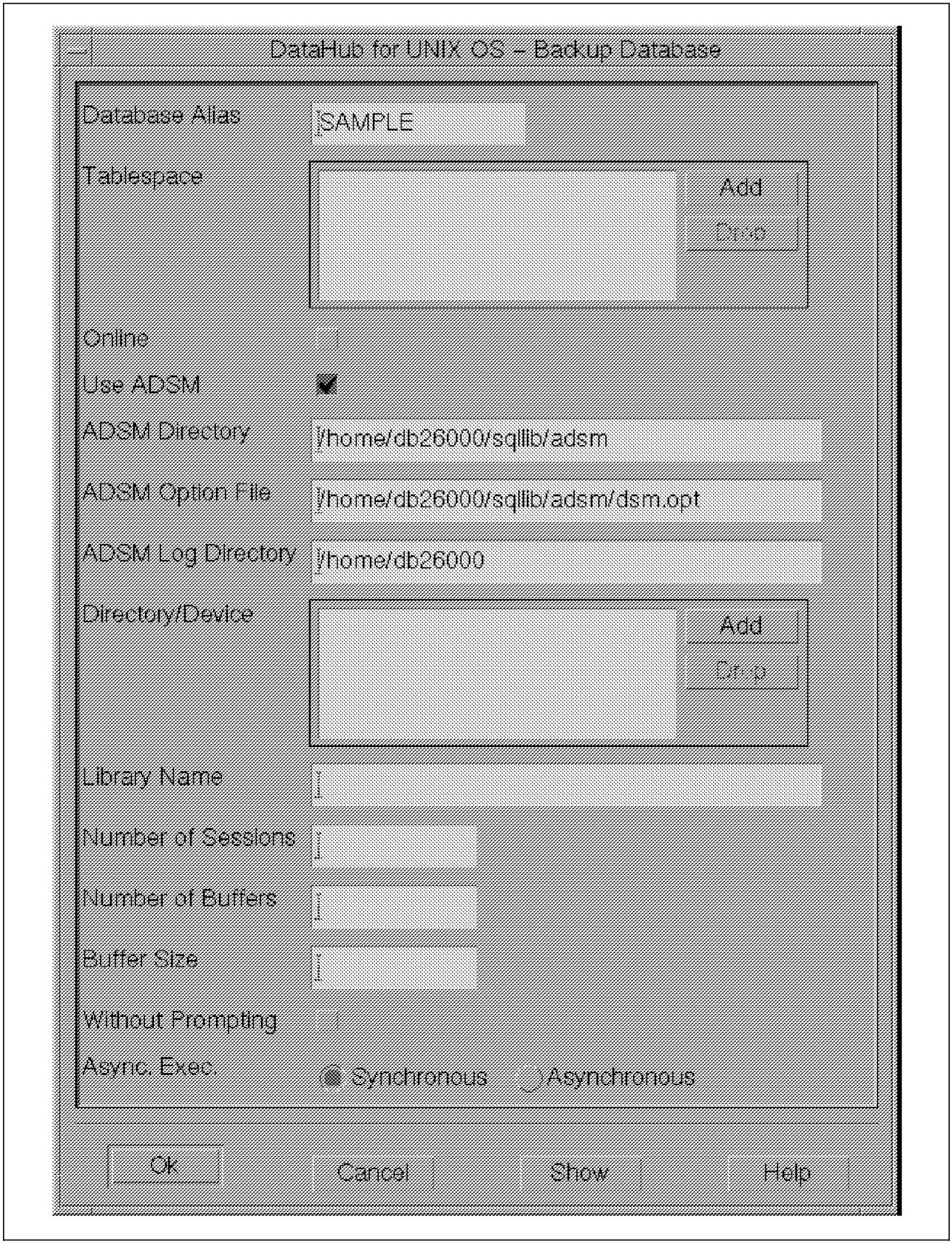


Figure 358. DataHub for UNIX OS Backup Database Window

Note: The following options are required to perform the backup. For information about other options refer to *Online Help*.

- The *Database Alias* is filled with the alias of your database.
- In the *Tablespace* option select tablespaces if you want to perform a tablespace backup. To back up a tablespace:
  a. Click on the Add button and the *Select One or More Tablespace Names* window appears.
  b. Select the desired tablespaces and click on OK to add them to the list.
- Enable the *online* option to do an Online backup. To perform an offline backup, ensure that there are no applications connected to the database you want to back up.
- Select the *Use ADSM* option to use ADSM for the backup. You must have previously customized your environment as described in Chapter 13, “ADSM and DB2/6000” on page 417. If the environment is already customized, the *ADSM Directory*, *ADSM Option File*, and the *ADSM Log Directory* fields are filled in automatically.
- *Number of Sessions* specifies the number of I/O sessions to be used. DB2/6000 can send data to ADSM using multiple sessions. See Chapter 13, “ADSM and DB2/6000” on page 417.
- *Number of Buffers* specifies the number of buffers to use during the backup. The default value is 2.
- *Buffer Size* specifies the size of the internal buffer used for the backup. The default value is 1024. If you have a large database, increase this value to improve performance.
- *Without Prompting* specifies that the backup runs unattended, and that any actions which normally require user intervention return an error message.
- *Synchronous* specifies that the backup runs in the foreground.
- *Asynchronous* specifies that the backup runs in the background.

5. Click on OK to begin the backup.
6. When the backup completes, a message window appears as shown in Figure 359.

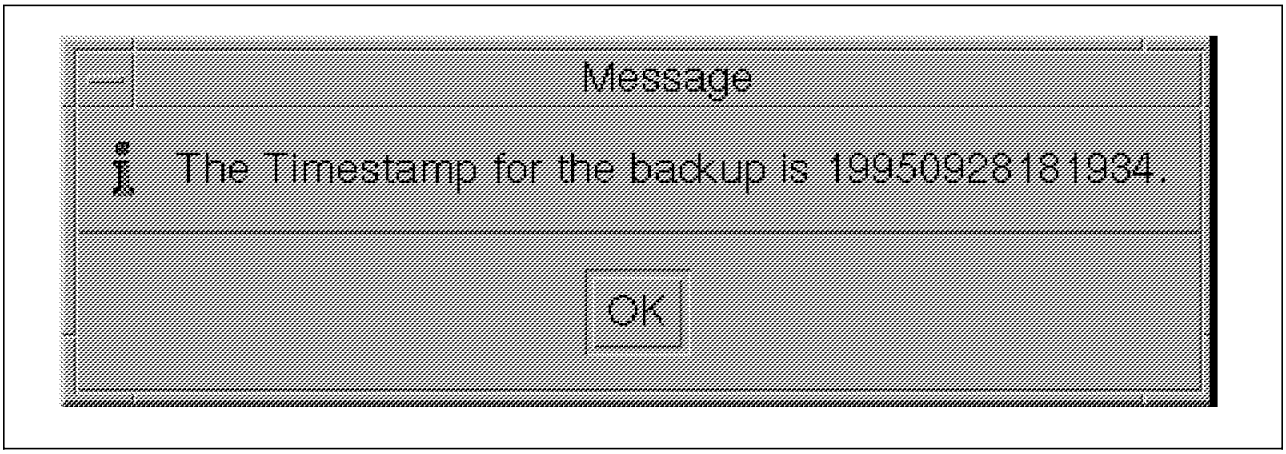


Figure 359. DataHub for UNIX OS Message Window for Backup

Write down the timestamp because it is used to uniquely identify the backup.

7. Click on *Cancel* to close the Backup Database window.

15.4 Restoring DB2/6000 with DataHub for UNIX OS

To restore a DB2/6000 database from DataHub for UNIX OS, perform the following steps (for this example we are using DB2/6000 Version 2):

1. Log in to your UNIX operating system as the instance owner for DB2/6000 (You must know the database backup timestamp.)
2. Select the database you want to recover. In our example we select the SAMPLE database.
3. Click on the Restore button in the Action window.

   The *Operating System Login for Database Operation* window appears. This window is used to provide verification for certain database, database manager, and operating system procedures. Every time a database (management) command is executed, a connection is set up. The *User* and *Password* combination is necessary for this connection. Enter the userid and password and click on OK.

   The Restore Database window appears as shown in Figure 360 on page 507.

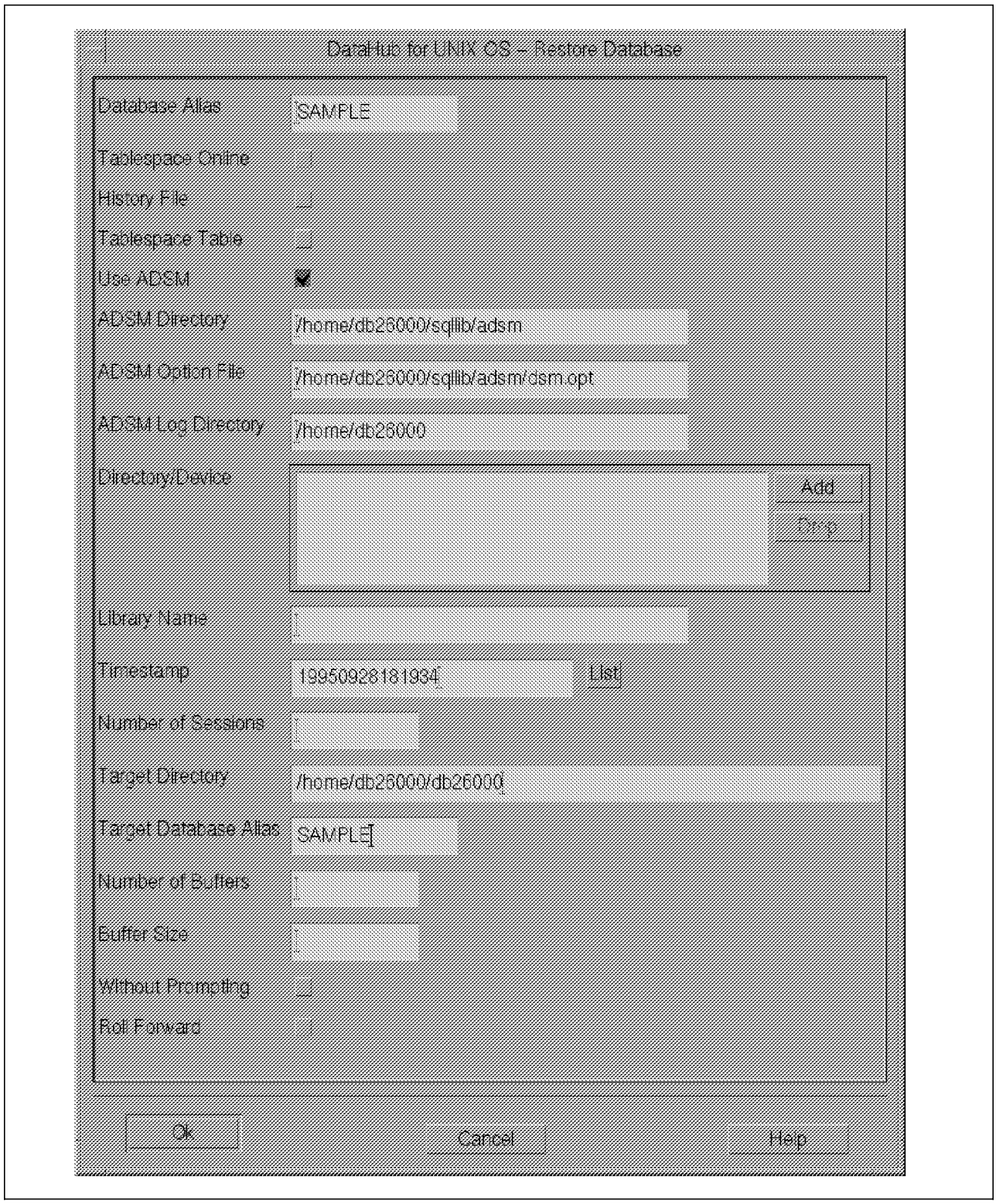


Figure 360. DataHub for UNIX OS Restore Database Window

Note: The following options are required to perform the recovery. For information about other options refer to the *Online Help*.

- The *Database Alias* is filled in with the alias for your database.
- *Tablespace Table* is not supported.

- Select the Use *ADSM* option to use ADSM for the restore. Customize your environment as described in Chapter 13, “ADSM and DB2/6000” on page 417. If the environment is already customized, the *ADSM Directory*, *ADSM Option File*, and the *ADSM Log Directory* fields are filled in automatically.
- *Timestamp* identifies the backup image to use for the restore. If you are using ADSM, the List button is not supported.
- *Number of Sessions* specifies the number of I/O sessions used with ADSM.
- *Target Directory* specifies the directory of the target database.
- *Target Database Alias* specifies the name of the database that is to be restored.
- *Number of Buffers* specifies the number of buffers to be used during the restore. The default is 2.
- *Buffer Size* specifies the size of the internal buffer used for the restore. The default value is 1024, but if you have large databases, increase this value to improve performance.
- *Without Prompting* specifies that the restore runs unattended, and that any actions which normally require user intervention will return an error message.
- *Roll Forward* specifies that you want to perform a roll-forward operation after the restore.

4. Click on OK to begin the restore process.
5. When the restore completes, a message window appears as shown in Figure 361.

Figure 361. DataHub for UNIX OS Message Window for Recovery

This warning refers to the SQL2539 warning, “Restoring to an existing database that is the same as the backup image database. The database file will be deleted. Do you want to continue? (y/n)..” DataHub for UNIX OS answers this warning; you do not have to enter anything.

6. Click on OK in the message window. A small message window informs you that the restore has completed successfully. Click on OK.

7. Click on *Cancel* to end the restore window.

For detailed information about backing up and restoring a DB2/6000 environment, refer to Chapter 13, “ADSM and DB2/6000” on page 417.

15.5 Backing Up Sybase with DataHub for UNIX OS

The only way you can back up Sybase from DataHub for UNIX OS is by using the Sybase dump utilities. To back up a Sybase database from DataHub for UNIX OS by using the *Dump Database* command, perform the following steps:

1. When the DataHub for UNIX OS main window appears, select the host and click on the Display button in the Action window.
2. Select Sybase Server from the *Select One or More* window and click on OK.
3. Select the server where your database is located and display it, using the Display button in the Action window.
4. Select Database from the *Select One or More* window and click OK.

   The *Managed Database Connect* window appears and asks for the *User Name* and *Password* for the Sybase database. (Note: The user name is the database administrator user name, not the operating system user name.) Click on Connect.
5. Select the database you want to back up. In our example we select the PUBS2 database.
6. Click on the Dump Database button in the Action window.

   The *Dump Database* window appears. See Figure 362 on page 510.

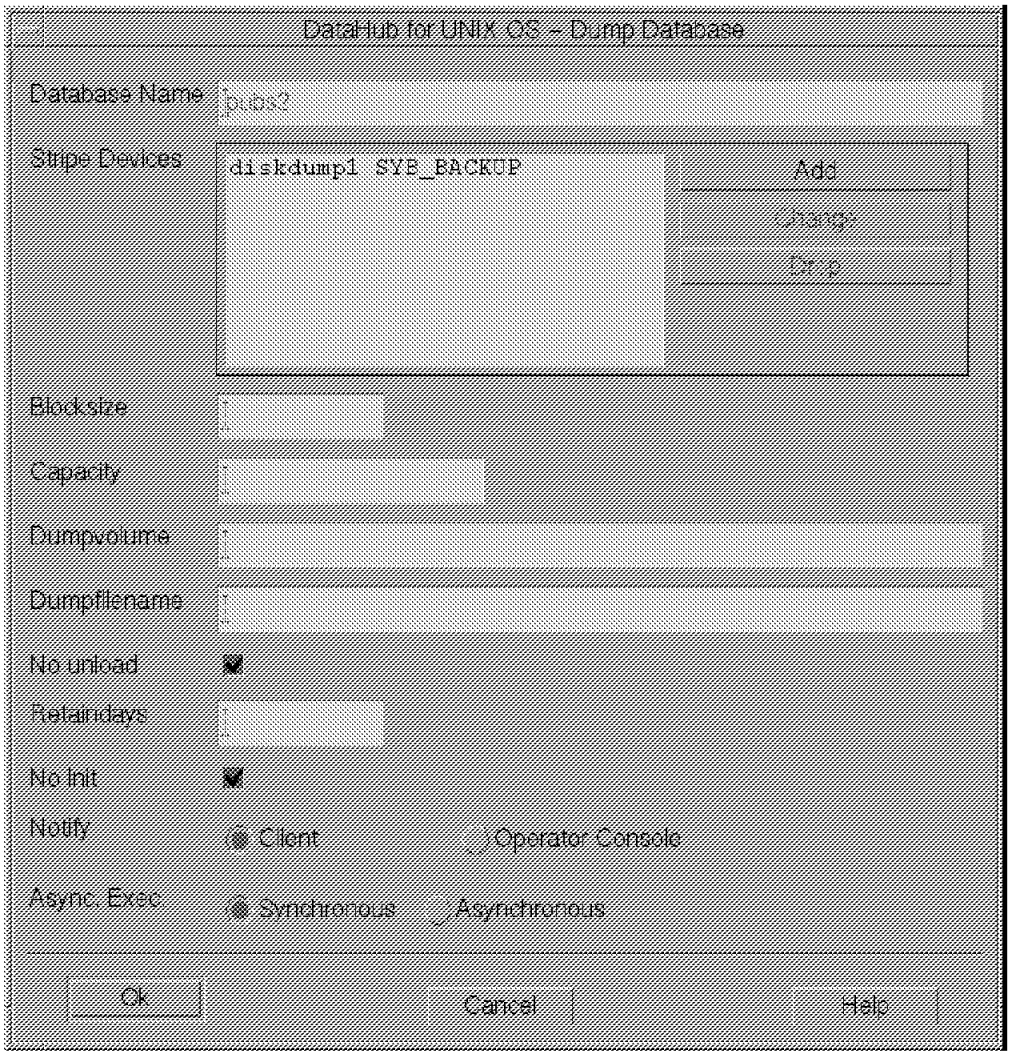


Figure 362. DataHub for UNIX OS Dump Database Window

Note: The following options are required to perform the dump. For information about other options refer to the *Online Help*.

- The *Database Name* is filled with the name of the selected database.
- For *Stripe Devices* you can specify the devices that will hold the dump data. You can define more than one stripe device to improve the performance of the backup. You must add at least one.

  To add a stripe device:

  a. Click on the Add button (not shown).

  b. Select the desired stripe device, using the *List* button, and click on OK.

  c. *Backup Server* specifies the name of the backup server that is running in the same machine as the SQL Server. You can find this server in the master.sysservers table. In our case it is SYB_BACKUP.

  d. Click on Add to add this stripe to the list. Then click on Close to end the add process.

- *Synchronous* specifies that the processes run in the foreground.
- *Asynchronous* specifies that the processes run in the background.

7. Click on OK to begin the dump.
8. The *Message* window appears and returns the status of your dump.
9. Click on *Cancel* to close the dump window.

If your transaction log files are stored on a different volume from your database files, you must also perform a dump transaction. The DataHub for UNIX OS window for *Dump Transaction* is similar to the *Dump Database* window; for more details about Sybase dump transactions, refer to 12.3.1, “Online Database Backup Using Dump Database and ADSM” on page 408.

15.6 Recovering Sybase with DataHub for UNIX OS

To recover a Sybase database from DataHub for UNIX OS by using the *Load Database* command, perform the following steps:

1. When the DataHub for UNIX OS main window appears, select the host and click on the Display button in the Action window.
2. Select Sybase Server from the *Select One or More* window and click on OK.
3. Select the server where your database is located and display it, using the Display button in the Action window.
4. Select Database from the *Select One or More* window and click on OK.

   The *Managed Database Connect* window appears and asks you for the *User Name* and *Password* for the Sybase database. Click on Connect.
5. Select the database you want to recover. In our example we select the PUBS2 database.
6. Click on the Load Database button.

   The *Load Database* window appears. See Figure 363 on page 512.

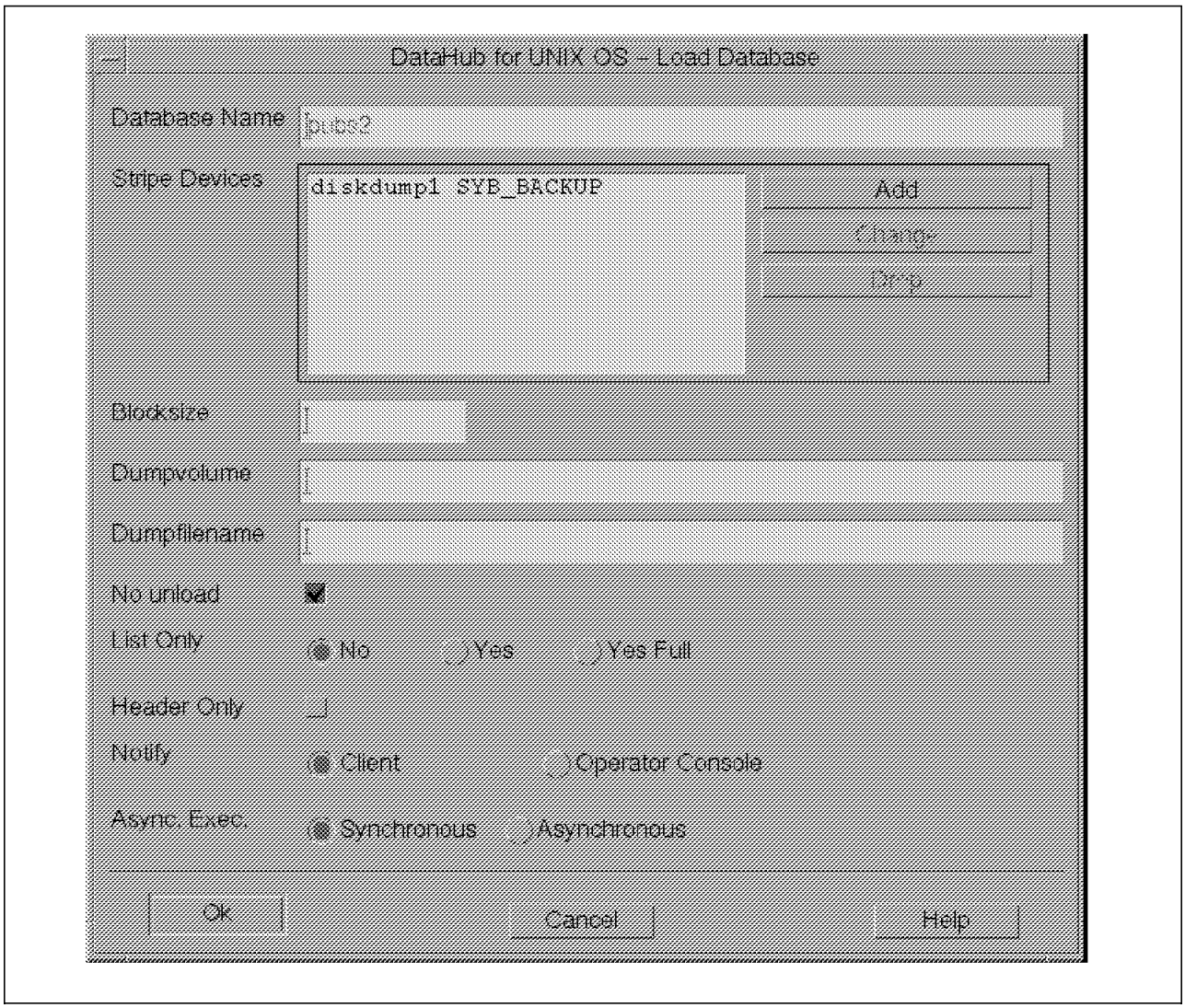


Figure 363. DataHub for UNIX OS Database Window

Note: The following options are required to perform the load. For information about other options refer to the *Online Help*.

- The *Database Name* is filled with the name of the selected database.
- For *Stripe Devices* you can specify the devices that will hold load data. You can define more than one stripe device.

  If you want to add a stripe device:

  a. Click on the Add button (not shown).

  b. Select the desired stripe device by clicking on the *List* button and then click on OK.

  c. *Backup Server* specifies the name of the backup server that is running in the same machine as the SQL Server. You can find this server in the master.sysservers table.

  d. Click on Add to add this stripe to the list. Then click on Close to end the add process.

- *Synchronous* specifies that the processes run in the foreground.

- *Asynchronous* specifies that the processes run in the background.

7. Click on OK to begin the load.
8. The *Message* window appears and returns the status of your load.
9. Click on *Cancel* to close the load window.

If you perform a dump transaction you must perform a load transaction. The *Load Transaction* window is similar to the DataHub for UNIX OS *Load Database* window; for more details on Sybase load transactions, refer to 12.3.1, “Online Database Backup Using Dump Database and ADSM” on page 408.

For more information about how to back up Sybase structures with ADSM, refer to Chapter 12, “ADSM and Sybase” on page 405.

15.7 Backing Up Oracle with DataHub for UNIX OS

Oracle does not provide its own backup utilities with the base product. At the time of writing, to back up an Oracle database with the utilities provided by DataHub for UNIX OS, all you can perform is an Oracle export. As described in Chapter 6, “Oracle Backup Basics” on page 81, export is not the method recommended for backing up an Oracle database. Nevertheless, here are the steps:

1. When the DataHub for UNIX OS window appears, select the host and click on the Display button in the Action window.
2. Select Database from the *Select One or More* window and click on OK.
3. Select the Oracle database you want to back up. In our example we select the ORFS database.
4. Click on the Export button in the Action window.

   The *Managed Database Connect* window appears. If you want to log in externally, specify *Identification External*. The *User* Name and *Password* combination is necessary for this connection. Enter the user name and password and click on OK.

   The Export window appears as shown in Figure 364 on page 514.

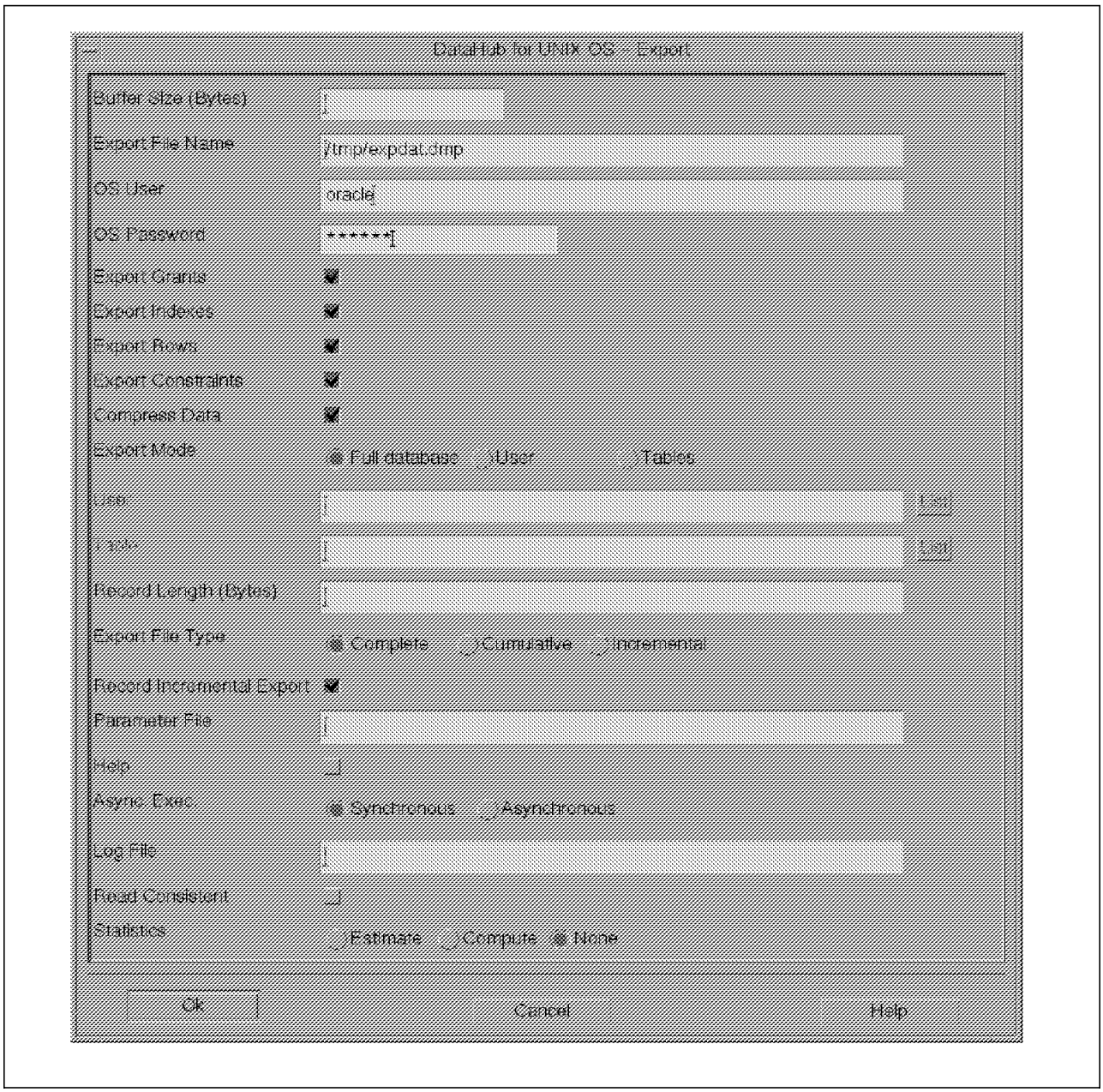


Figure 364. DataHub for UNIX OS Export Window

Note: The following options are required to perform the export. For information about other options refer to the *Online Help*.

- *Export File Name* is filled with the name of the file that will hold the exported data. You can change this name.
- *OS User* specifies the name of the user in the operating system that will perform the export.
- *OS Password* specifies the password of the user in the operating system that will perform the export.
- The following Export options are enabled:
  - *Export Grants* if you want to export the grants for the database.
  - *Export Indexes* if you want to export the indexes for the database.

- *Export Rows* if you want to export the rows of the table data.
- *Export Constraints* if you want to export the table constraints.
- *Compress Data* if you want the table data, when imported later, to be compressed into one extent.

- *Export Mode*
  - *Full database*, to export the entire database.
  - *User*, to export the tables of a certain user.

    You can use the List option to display the user list.
  - *Tables*, to specify predetermined tables.

    You can use the List option to display the tables list.
- *Async. Exec.*
  - *Synchronous* specifies that the processes run in the foreground.
  - *Asynchronous* specifies that the processes run in the background.

5. Click OK to begin the export.
6. When the export completes, a message window appears as shown Figure 365.

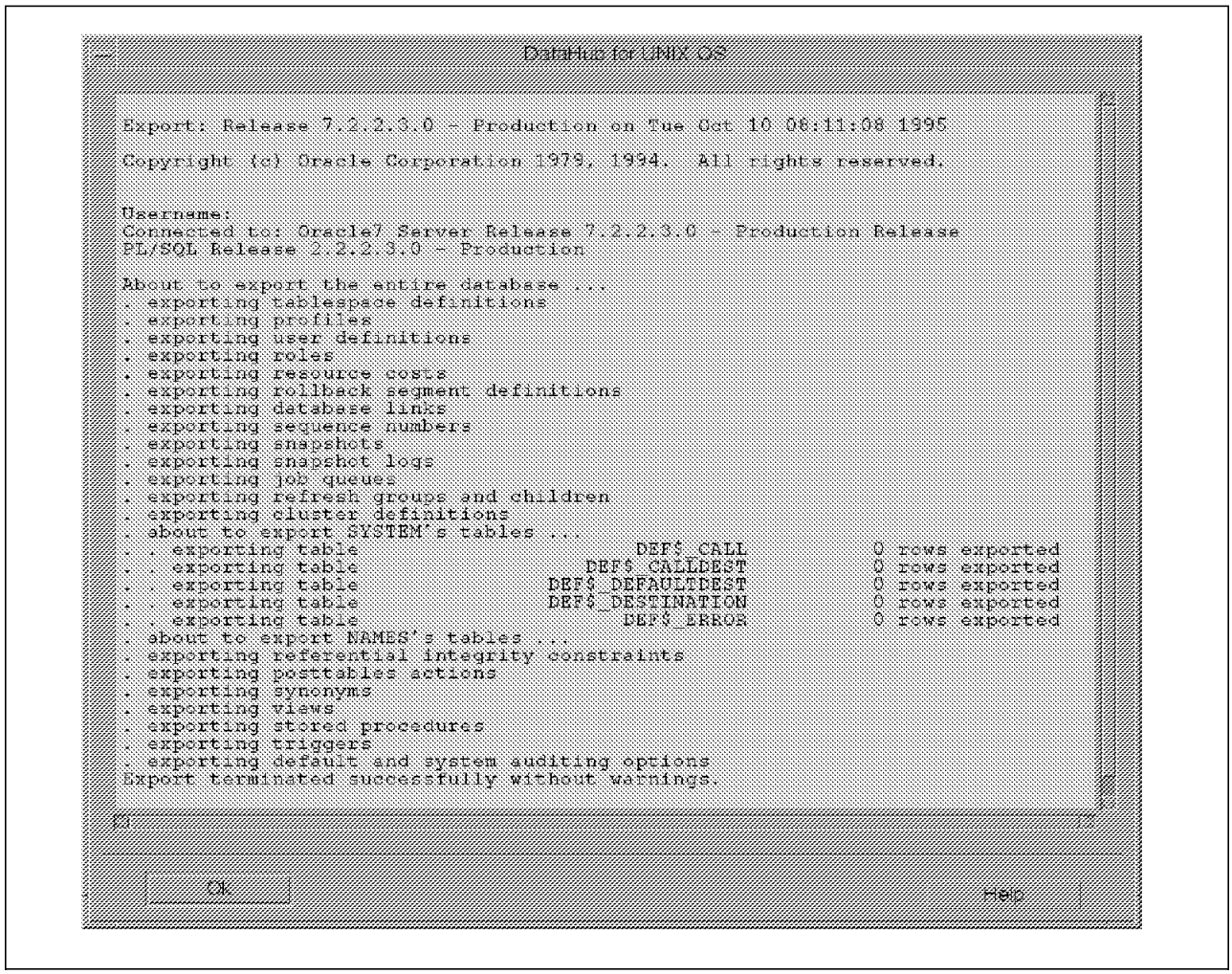


Figure 365. DataHub for UNIX OS Message Window for Export

7. Click on *Cancel* to close the export window.

15.8 Recovering Oracle with DataHub for UNIX OS

To recover an Oracle database from DataHub you must use the import utility:

1. When the DataHub for UNIX OS main window appears, select the host and click on the Display button in the Action window.
2. Select Database from the *Select One or More* window and click on OK.
3. Select the Oracle database you want to recover. In our example we select the ORFS database.
4. Click on the Import button in the Action window.

   The *Managed Database Connect* window appears. If you want to log in externally, specify *Identification External*. The *User* Name and *Password* combination is necessary for this connection. Enter the user name and password and click on OK.

   The Import window appears as shown in Figure 366 on page 517.

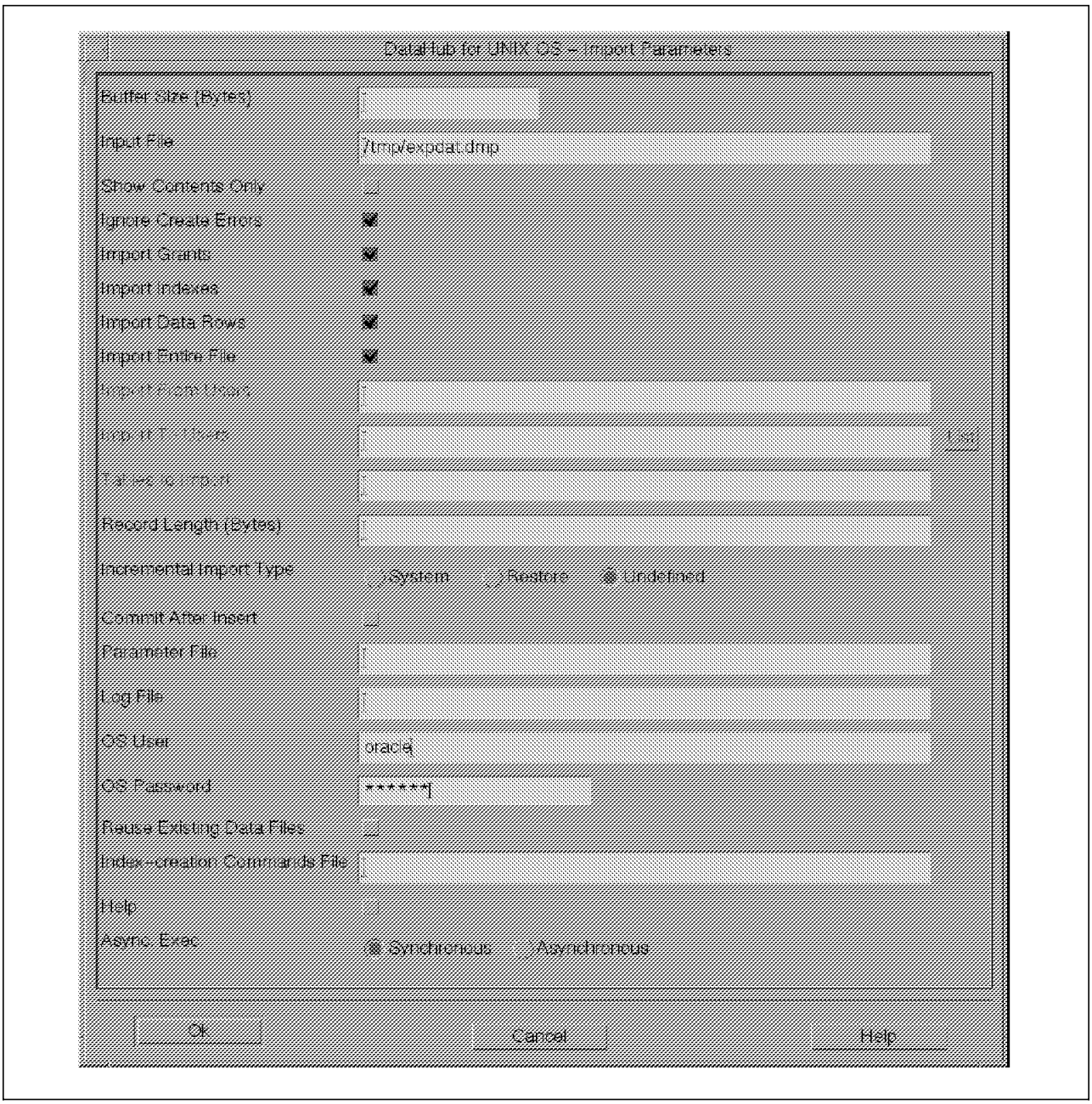


Figure 366. DataHub for UNIX OS Import Parameters

Note: The following options are required to perform the import. For information about other options refer to the *Online Help*.

- *Input File* is filled with the name of the file that contains the data to import. You can change this name.
- The following import options are enabled.
  - *Ignore Create Errors* to avoid the message “object already exists.” It ignores the error and continues with the import.
  - *Import Grants* if you want to import the grants for the database.
  - *Import Indexes* if you want to import the indexes for the database.
  - *Import Data Rows* if you want to import the rows of the table data.

- *Import Entire File* to import the contents of the file. If you enable this option, the *Incremental Import Type* options are enabled.

- Incremental Import Type
  - *System* imports the most recent version of system objects. User objects are not imported.
  - *Restore* imports all user objects and data that have changed since the last export.
  - *Undefined* does not drop current objects in favor of imported objects. Undefined is the default.
- *OS User* specifies the name of the user that will be the owner of the files created by the import synchronous operations.
- *OS Password* specifies the password of the user for the operating system.
- *Synchronous* specifies that the processes run in the foreground.
- *Asynchronous* specifies that the processes run in the background.

5. Click on OK to begin the import.
6. The *Message* window appears with the message shown in Figure 367.

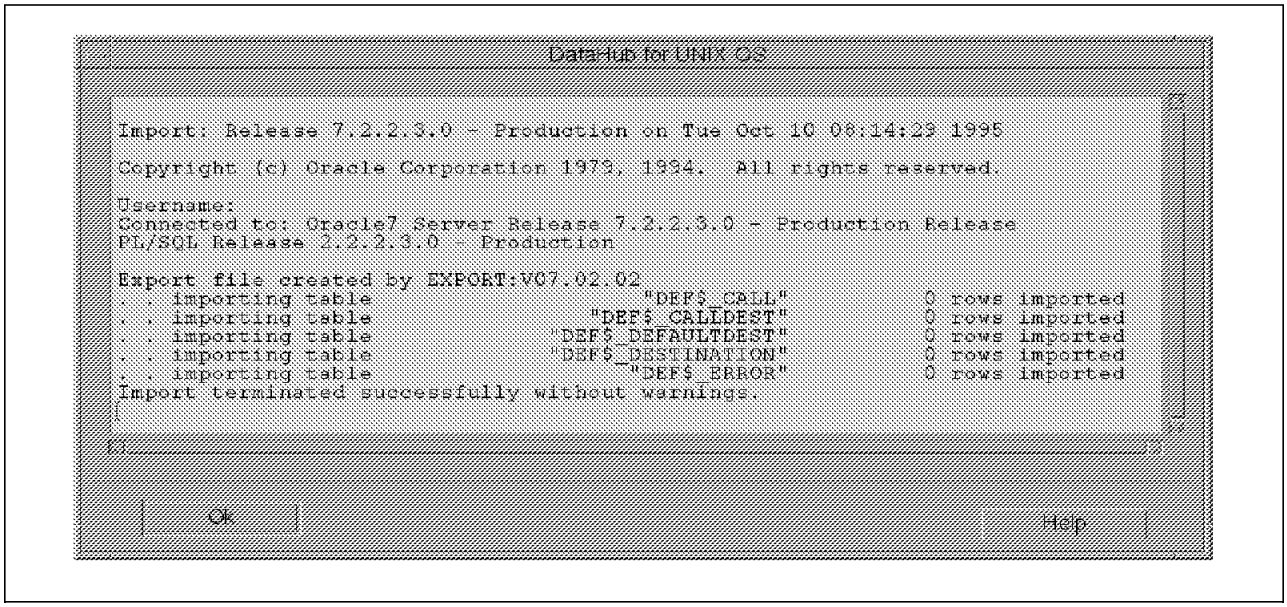


Figure 367. DataHub for UNIX OS Import Message Window

7. Click on *Cancel* to close the Import window.

You can also perform commands for tablespaces, such as *Online* and *Offline*, or run procedures such as *Begin Backup* or *End Backup*. Display the objects from your Oracle database and select the tablespace options (they appear on the left side of the DataHub for UNIX OS window). For more information about backing up Oracle structures refer to Chapter 6, “Oracle Backup Basics” on page 81.

15.9 Backing Up DB2/6000 with DataHub for OS/2

To back up a DB2/6000 database from DataHub for OS/2, perform the following steps (for this example we are using DB2/6000 Version 2):

1. Start DataHub for OS/2 by double-clicking on the DataHub icon or, from an OS/2 window, type datahub.
2. When the DataHub for OS/2 window appears, select the DB2/6000 managed host. From the *Actions* pull-down menu select *Display...*.
3. The *Display Related Objects* window appears. The RDB (relational databases) option is already selected. Click on Display.
4. Select the database you want to back up. In our example we select the SAMPLE_1 database.
5. From the Actions pull-down menu select *Utilities* and then *Backup...*

   The *Backup Database* window appears as shown in Figure 368.

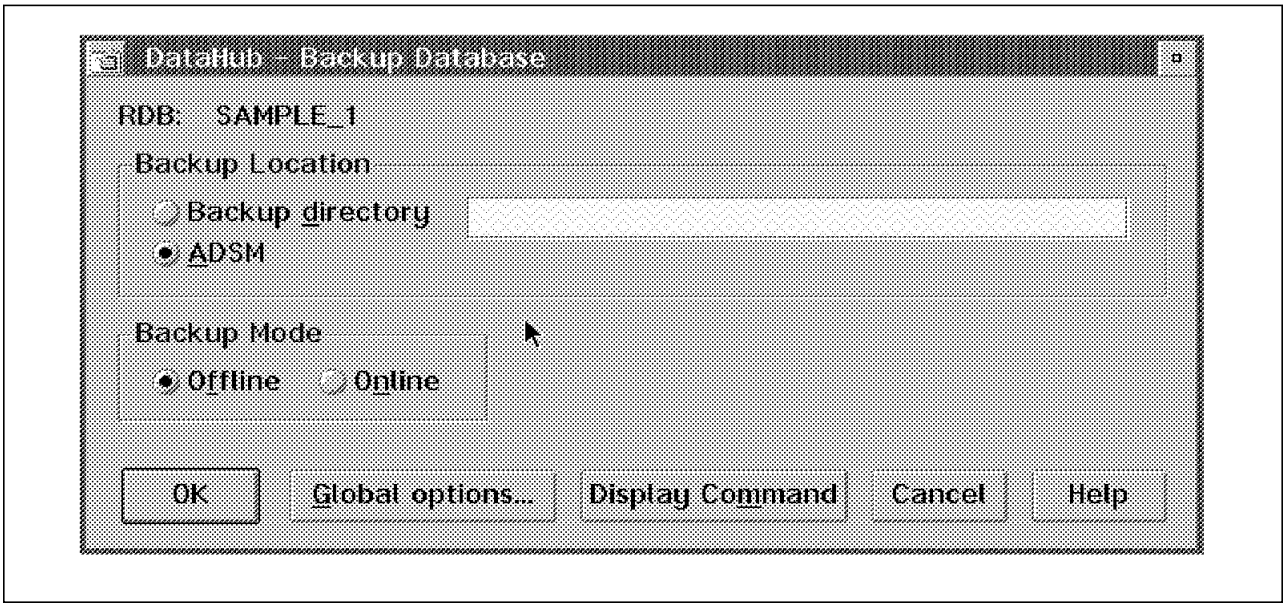


Figure 368. DataHub for OS/2 Backup Database Window

- *RDB* is filled with the selected alias for your database.
- *Backup Location* shows two options: *Backup directory* and *ADSM*. Select ADSM.
- *Backup Mode* asks for *Online* or *Offline*. If you specify Offline, you must ensure that there are no applications connected to the database; you may have database connections of which you are not aware. When you display some objects in the tree they may create a connection to the database. If you specify Online, you must ensure that the Database Manager Configuration has enabled either the Log Retain Enable or User Exit option and that there is not a Backup pending process.
- To see the DataHub for OS/2 generated command, click on the *Display Command* button.
- To generate a report on the status of your command, click on the *Global options...* button.

6. Click on OK to begin the back up.
7. The *Backup Complete* window appears and displays the message shown in Figure 369.

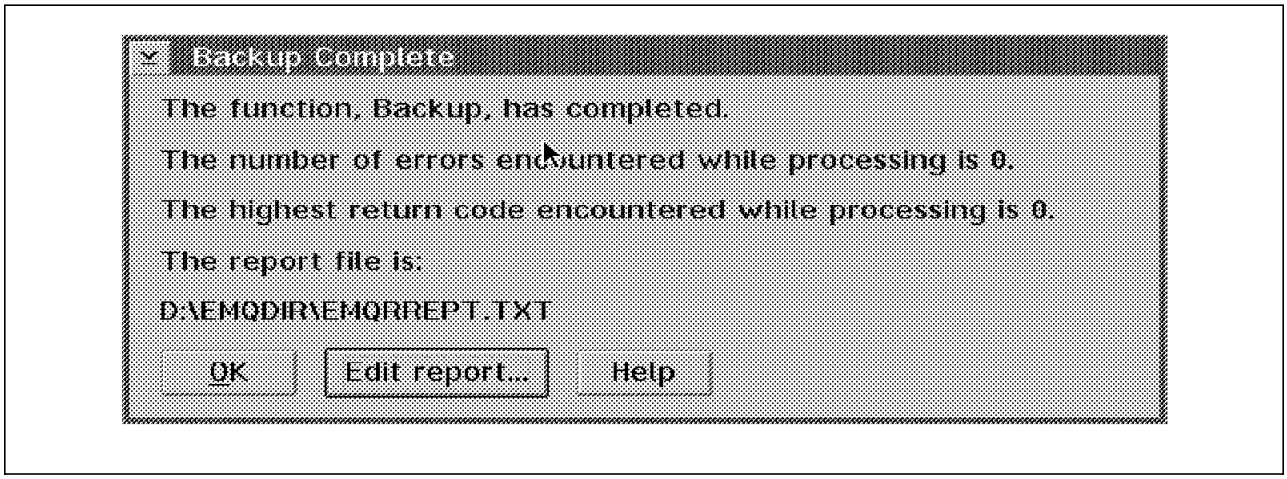


Figure 369. DataHub for OS/2 Backup Message Window

If there are errors in this window, click on the Edit report... button to find the error.

15.10 Recovering DB2/6000 with DataHub for OS/2

To recover a DB2/6000 database from DataHub for OS/2, perform the following steps (for this example we are using DB2/6000 Version 2):

1. Start DataHub for OS/2 by double-clicking on the DataHub icon, or, from an OS/window, type datahub.
2. When the DataHub for OS/2 window appears, select the DB2/6000 managed host. From the *Actions* pull-down menu select *Display...*.
3. The *Display Related Objects* window appears. The RDB (relational databases) option is already selected. Click on Display.
4. Select the database you want to back up. In our example we select the SAMPLE_1 database.
5. From the Actions pull-down menu select *Utilities* and then *Recover...*.

   The *Recover Database* window appears as shown in Figure 370 on page 521.

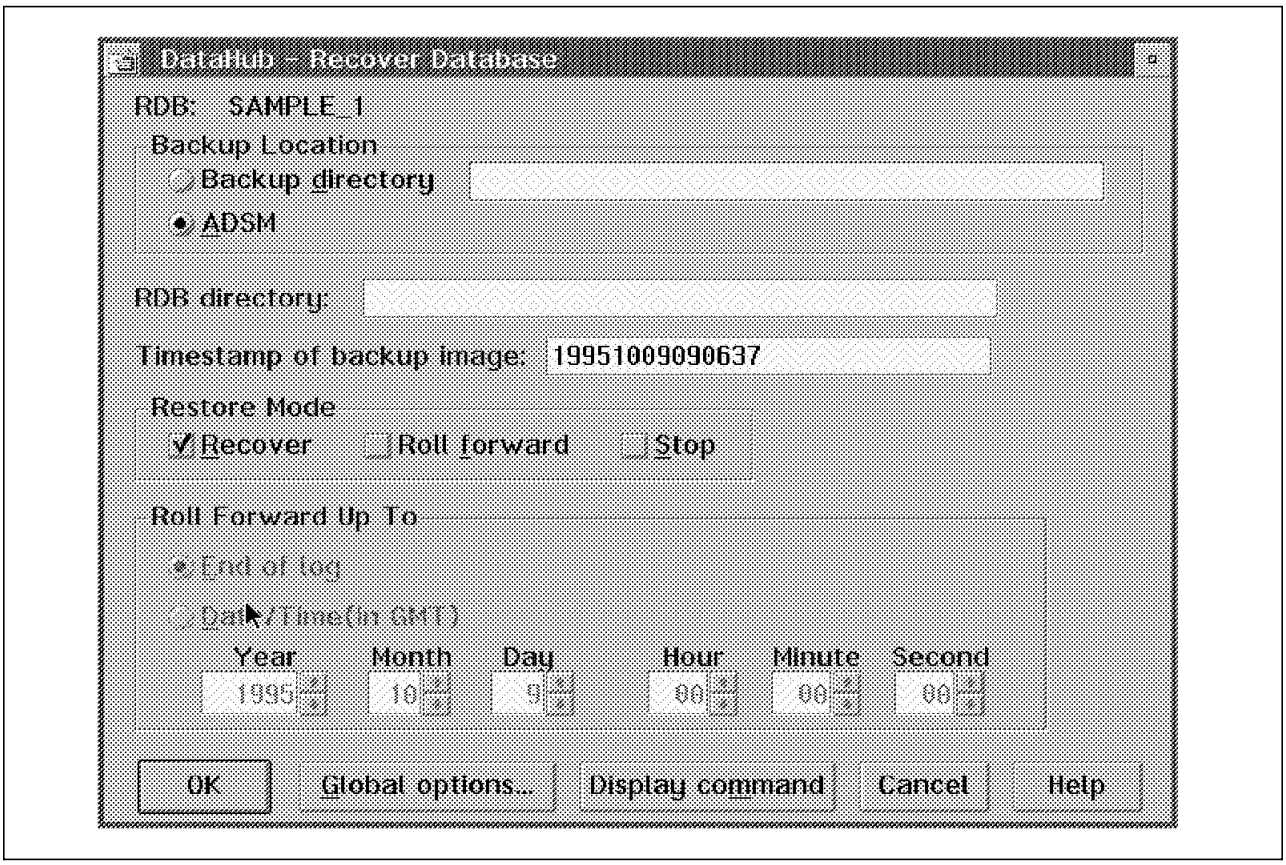


Figure 370. DataHub for OS/2 Recover Database Window

The Figure 370 contains the following fields:

- *RDB* is filled with the selected alias for your database.
- *Backup Location* shows two options: *Backup directory* and *ADSM*. Select ADSM.
- In the *Timestamp of backup Image* field enter the timestamp of the backup you want to recover.
- *Restore Mode* can be:
  - *Recover*, which recovers the database to the state it was in when it was backed up.
  - *Roll forward*, which applies the changes in the log file. You must ensure that your database manager configuration has enabled either the Log Retain Enable or User Exit Option.
  - *Stop*, which specifies that the database can be ready to use after the recover; Stop applies only when using the Roll Forward Up To option.
- *Roll Forward Up To*

  -*End of log* specifies that roll forward is applied up to the last active log file.

  -*Date/Time(in GMT)* specifies that roll forward is applied to a certain point in time.
- To see the DataHub for OS/2 generated command, click on the *Display command* button.

- To generate a report on the status of your command, click on the *Global options...* button.
- Click on OK to begin the restore.

6. The *Recover Complete* window appears and displays the message shown in Figure 371.

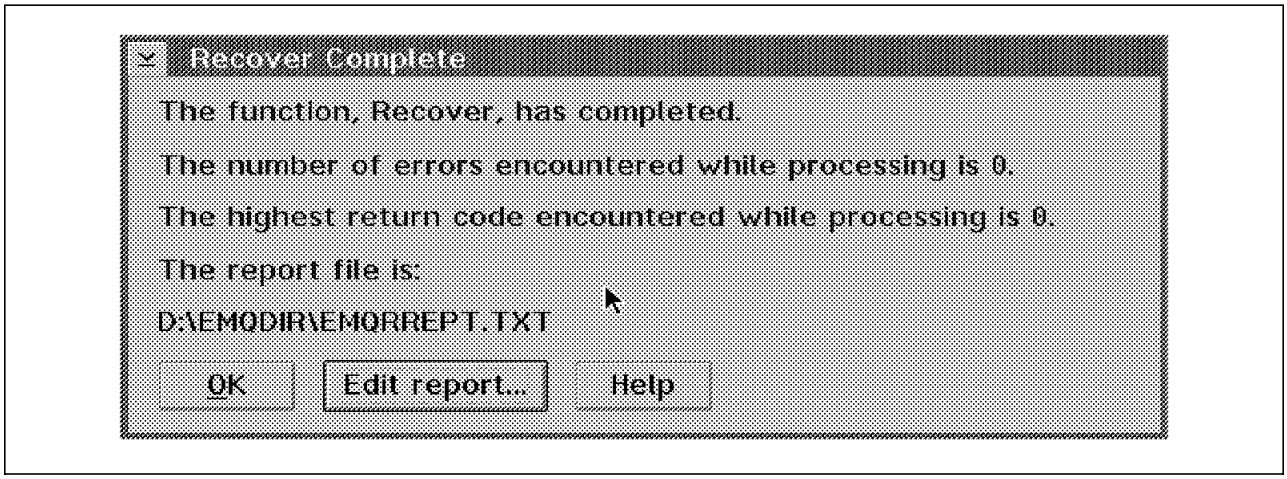


Figure 371. DataHub for OS/2 Recover Message Window

If there are errors in this window, click on the Edit report... button to find the error.

15.11 Backing up DB2/2 with DataHub for OS/2

As mentioned in the Chapter 14, “ADSM and DB2/2” on page 465, backing up a DB2/2 Version 1 database directly to ADSM is currently not supported. Although DB2/2 Version 2 backs up directly to ADSM, you cannot do the backup from DataHub for OS/2.

If you have DB2/2 Version 1, you can do the backup from DataHub for OS/2 if you use the user exit code as described in 14.3.2, “Offline Backup Using the DB2/2 V1 User Exit Integrated with ADSM” on page 474. This method is not practical for Version 2, because the user exit program has changed and supports only archive and retrieve of logs, not backup and restore of data.

Appendix A. Database Configuration Files

This appendix presents the configuration files used for INFORMIX-OnLine Versions 5 and 6, Sybase, DB2/6000, and DB2/2.

A.1 INFORMIX-OnLine Version 5 Configuration Files

A.1.1 File tbconfig for Database Server Operating on File System Files

```
#**************************************************************************
#
#                  INFORMIX SOFTWARE, INC.
#
#  Title:  tbconfig.std
#  Sccsid:      @(#)tbconfig.std     9.1.2.1    2/4/93  12:01:23
#  Description: INFORMIX-OnLine Configuration Parameters
#
#**************************************************************************

# Root Dbspace Configuration

ROOTNAME        rootdbs         # Root dbspace name
ROOTPATH        /usr/informix5/dbspaces/inf5rootdbs
                                # Path for device containing root dbspace
ROOTOFFSET      0               # Offset of root dbspace into device (Kbytes)
ROOTSIZE        20000           # Size of root dbspace (Kbytes)

# Disk Mirroring Configuration

MIRROR          0               # Mirroring flag (Yes = 1, No = 0)
MIRRORPATH                      # Path for device containing root dbspace mirror
MIRROROFFSET    0               # Offset into mirror device (Kbytes)

# Physical Log Configuration

PHYSDBS         rootdbs         # Name of dbspace that contains physical log
PHYSFILE        2000            # Physical log file size (Kbytes)

# Logical Log Configuration

LOGFILES        6               # Number of logical log files
LOGSIZE         1000            # Size of each logical log file (Kbytes)

# Message Files

MSGPATH         /usr/informix5/online.log # OnLine message log pathname
CONSOLE         /dev/console    # System console message pathname

# Archive Tape Device

TAPEDEV         /home/informix5/tapedev # Archive tape device pathname
TAPEBLK         16              # Archive tape block size (Kbytes)
TAPESIZE        50000           # Max amount of data to put on tape (Kbytes)

# Logical Log Backup Tape Device
```

```
LTAPEDEV        /home/informix5/ltapedev # Logical log tape device pathname
LTAPEBLK        16              # Logical log tape block size (Kbytes)
LTAPESIZE       50000           # Max amount of data to put on log tape (Kbytes)

# Identification Parameters

SERVERNUM       0               # Unique id associated with this OnLine instance
DBSERVERNAME    ONLINE          # Unique name of this OnLine instance

# Shared Memory Parameters

RESIDENT        0               # Forced residency flag (Yes = 1, No = 0)
USERS           20              # Maximum number of concurrent user processes
TRANSACTIONS    20              # Maximum number of concurrent transactions
LOCKS           2000            # Maximum number of locks
BUFFERS         200             # Maximum number of shared memory buffers
TBLSPACES       200             # Maximum number of active tblspaces
CHUNKS          8               # Maximum number of chunks
DBSPACES        8               # Maximum number of dbspaces and blobspaces
PHYSBUFF        128             # Size of physical log buffers (Kbytes)
LOGBUFF         128             # Size of logical log buffers (Kbytes)
LOGSMAX         12              # Maximum number of logical log files
CLEANERS        1               # Number of page-cleaner processes
SHMBASE         0x30000000        # Shared memory base address
CKPTINTVL       300             # Checkpoint interval (in seconds)
LRUS            8               # Number of LRU queues
LRU_MAX_DIRTY   60              # LRU modified begin-cleaning limit (percent)
LRU_MIN_DIRTY   50              # LRU modified end-cleaning limit (percent)
LTXHWM          80              # Long TX high-water mark (percent)
LTXEHWM         90              # Long TX exclusive high-water mark (percent)

# Machine- and Product-Specific Parameters

DYNSHMSZ        0               # Dynamic shared memory size (Kbytes)
GTRID_CMP_SZ    32              # Number of bytes to use in GTRID comparision
DEADLOCK_TIMEOUT 60              # Max time to wait for lock in distributed env.
TXTIMEOUT       300             # Transaction timeout for I-STAR (in seconds)
SPINCNT         0               # No. of times process tries for latch
STAGEBLOB                       # Reserved for INFORMIX-OnLine/Optical

# System Page Size

BUFFSIZE        4096            # Page size (do not change!)
```

A.1.2 dbspaces for Database Server Operating on File System Files

```
$ tbstat -d

RSAM Version 5.02.UC3   -- On-Line -- Up 00:00:10 -- 1728 Kbytes

Dbspaces
address  number   flags    fchunk   nchunks  flags    owner    name
3000a4ac 1        1        1        1        N        informix rootdbs
3000a4dc 2        1        2        1        N        informix inf5fs
3000a50c 3        1        3        1        N        informix inf5fs2
 3 active, 8 total
```

```
Chunks
address  chk/dbs offset size free flags pathname
30009fec 1   1   0      5000 1613  PO-  /usr/informix5/dbspaces/inf5rootdbs
3000a084 2   2   0      5000 3609  PO-  /usr/informix5/dbspaces/inf5fs
3000a11c 3   3   0      5000 4892  PO-  /usr/informix5/dbspaces/inf5fs2
 3 active, 8 total
```

A.1.3 File tbconfig for Database Server Operating on Raw Devices

```
#**************************************************************************
#
#                  INFORMIX SOFTWARE, INC.
#
#  Title:  tbconfig.std
#  Sccsid:      @(#)tbconfig.std     9.1.2.1    2/4/93  12:01:23
#  Description: INFORMIX-OnLine Configuration Parameters
#
#**************************************************************************

# Root Dbspace Configuration

ROOTNAME        rootdbs         # Root dbspace name
ROOTPATH        /dev/rinf5rootdbs # Path for device containing root dbspace
ROOTOFFSET      0               # Offset of root dbspace into device (Kbytes)
ROOTSIZE        20000           # Size of root dbspace (Kbytes)

# Disk Mirroring Configuration

MIRROR          0               # Mirroring flag (Yes = 1, No = 0)
MIRRORPATH                      # Path for device containing root dbspace mirror
MIRROROFFSET    0               # Offset into mirror device (Kbytes)

# Physical Log Configuration

PHYSDBS         rootdbs         # Name of dbspace that contains physical log
PHYSFILE        2000            # Physical log file size (Kbytes)

# Logical Log Configuration

LOGFILES        6               # Number of logical log files
LOGSIZE         1000            # Size of each logical log file (Kbytes)

# Message Files

MSGPATH         /usr/informix5/online.log # OnLine message log pathname
CONSOLE         /dev/console    # System console message pathname

# Archive Tape Device

TAPEDEV         /home/informix5/tapedev # Archive tape device pathname
TAPEBLK         16              # Archive tape block size (Kbytes)
TAPESIZE        50000           # Max amount of data to put on tape (Kbytes)

# Logical Log Backup Tape Device

LTAPEDEV        /home/informix5/ltapedev # Logical log tape device pathname
LTAPEBLK        16              # Logical log tape block size (Kbytes)
LTAPESIZE       50000           # Max amount of data to put on log tape (Kbytes)
```

```
# Identification Parameters

SERVERNUM       0               # Unique id associated with this OnLine instance
DBSERVERNAME    ONLINE          # Unique name of this OnLine instance

# Shared Memory Parameters

RESIDENT        0               # Forced residency flag (Yes = 1, No = 0)
USERS           20              # Maximum number of concurrent user processes
TRANSACTIONS    20              # Maximum number of concurrent transactions
LOCKS           2000            # Maximum number of locks
BUFFERS         200             # Maximum number of shared memory buffers
TBLSPACES       200             # Maximum number of active tblspaces
CHUNKS          8               # Maximum number of chunks
DBSPACES        8               # Maximum number of dbspaces and blobspaces
PHYSBUFF        128             # Size of physical log buffers (Kbytes)
LOGBUFF         128             # Size of logical log buffers (Kbytes)
LOGSMAX         12              # Maximum number of logical log files
CLEANERS        1               # Number of page-cleaner processes
SHMBASE         0x30000000        # Shared memory base address
CKPTINTVL       300             # Checkpoint interval (in seconds)
LRUS            8               # Number of LRU queues
LRU_MAX_DIRTY   60              # LRU modified begin-cleaning limit (percent)
LRU_MIN_DIRTY   50              # LRU modified end-cleaning limit (percent)
LTXHWM          80              # Long TX high-water mark (percent)
LTXEHWM         90              # Long TX exclusive high-water mark (percent)

# Machine- and Product-Specific Parameters

DYNSHMSZ        0               # Dynamic shared memory size (Kbytes)
GTRID_CMP_SZ    32              # Number of bytes to use in GTRID comparision
DEADLOCK_TIMEOUT 60              # Max time to wait for lock in distributed env.
TXTIMEOUT       300             # Transaction timeout for I-STAR (in seconds)
SPINCNT         0               # No. of times process tries for latch
STAGEBLOB                       # Reserved for INFORMIX-OnLine/Optical

# System Page Size

BUFFSIZE        4096            # Page size (do not change!)
```

A.1.4 dbspaces for Database Server Operating on Raw Devices

```
$ tbstat -d

RSAM Version 5.02.UC3   -- On-Line -- Up 01:41:38 -- 1728 Kbytes

Dbspaces
address  number   flags    fchunk   nchunks  flags    owner    name
3000a4ac 1        1        1        1        N        informix rootdbs
3000a4dc 2        1        2        1        N        informix inf5rd
 2 active, 8 total

Chunks
address  chk/dbs offset   size     free     bpages   flags pathname
30009fec 1   1   0        5000     1828              PO-   /dev/rinf5rootdbs
3000a084 2   2   0        5000     3609              PO-   /dev/rinf5rd
 2 active, 8 total
```

A.2 INFORMIX-OnLine Version 6 Configuration Files

A.2.1 File onconfig for Database Server Operating on File System Files

```
#**************************************************************************
#
#                  INFORMIX SOFTWARE, INC.
#
#  Title:  onconfig.std
#  Sccsid:      @(#)onconfig.std     9.24  9/30/93    10:04:32
#  Description: INFORMIX-OnLine Configuration Parameters
#
#**************************************************************************

# Root Dbspace Configuration

ROOTNAME        rootdbs         # Root dbspace name
ROOTPATH        /usr/informix6/dbspaces/inf6rootdbs
                                # Path for device containing root dbspace
ROOTOFFSET      0               # Offset of root dbspace into device (Kbytes)
ROOTSIZE        20000           # Size of root dbspace (Kbytes)

# Disk Mirroring Configuration Parameters

MIRROR          1               # Mirroring flag (Yes = 1, No = 0)
MIRRORPATH
                                # Path for device containing mirrored root
MIRROROFFSET    0               # Offset into mirrored device (Kbytes)

# Physical Log Configuration

PHYSDBS         rootdbs         # Location (dbspace) of physical log
PHYSFILE        2000            # Physical log file size (Kbytes)

# Logical Log Configuration

LOGFILES        6               # Number of logical log files
LOGSIZE         1000            # Logical log size (Kbytes)

# Message Files

MSGPATH         /usr/informix6/online.log # System message log file path
CONSOLE         /dev/console    # System console message path

# System Archive Tape Device

TAPEDEV         /home/informix6/tapedev # Tape device path
TAPEBLK         16              # Tape block size (Kbytes)
TAPESIZE        50000           # Maximum amount of data to put on tape (Kbytes)

# Log Archive Tape Device

LTAPEDEV        /home/informix6/ltapedev # Log tape device path
LTAPEBLK        16              # Log tape block size (Kbytes)
LTAPESIZE       10240           # Max amount of data to put on log tape (Kbytes)
```

```
# Optical

STAGEBLOB                       # INFORMIX-OnLine/Optical staging area

# System Configuration

SERVERNUM       1               # Unique id corresponding to a OnLine instance
DBSERVERNAME    balticshm       # Name of default database server
DBSERVERALIASES                 # List of alternate dbservernames
DEADLOCK_TIMEOUT 60              # Max time to wait of lock in distributed env.
RESIDENT        0               # Forced residency flag (Yes = 1, No = 0)

MULTIPROCESSOR  0               # 0 for single-processor, 1 for multi-processor
NUMCPUVPS       1               # Number of user (cpu) vps
SINGLE_CPU_VP   0               # If non-zero, limit number of cpu vps to one

NOAGE           0               # Process aging
AFF_SPROC       0               # Affinity start processor
AFF_NPROCS      0               # Affinity number of processors

# Shared Memory Parameters

USERTHREADS     20              # Maximum number of concurrent threads
TRANSACTIONS    20              # Maximum number of concurrent transactions
LOCKS           2000            # Maximum number of locks
BUFFERS         200             # Maximum number of shared buffers
TBLSPACES       200             # Maximum number of open tblspaces
CHUNKS          8               # Maximum number of chunks
NUMAIOVPS       2               # Number of IO vps
DBSPACES        8               # Maximum number of dbspaces
PHYSBUFF        128             # Physical log buffer size (Kbytes)
LOGBUFF         128             # Logical log buffer size (Kbytes)
LOGSMAX         6               # Maximum number of logical log files
CLEANERS        1               # Number of buffer cleaner processes
SHMBASE         0x30000000        # Shared memory base address
SHMVIRTSIZE     8000            # initial virtual shared memory segment size
SHMADD          8192            # Size of new shared memory segments (Kbytes)
SHMTOTAL        0               # Total shared memory (Kbytes). 0=>unlimited
CKPTINTVL       300             # Check point interval (in sec)
LRUS            8               # Number of LRU queues
LRU_MAX_DIRTY   60              # LRU percent dirty begin cleaning limit
LRU_MIN_DIRTY   50              # LRU percent dirty end cleaning limit
LTXHWM          80              # Long transaction high water mark percentage
LTXEHWM         90              # Long transaction high water mark (exclusive)
TXTIMEOUT       0x12c             # Transaction timeout (in sec)
STACKSIZE       32              # Stack size (Kbytes)

# System Page Size
# BUFFSIZE - OnLine no longer supports this configuration parameter.
#            To determine the page size used by OnLine on your platform
#            see the last line of output from the command, 'onstat -b'.


# Recovery Variables
# OFF_RECVRY_THREADS:
# Number of parallel worker threads during fast recovery or an offline restore.
# ON_RECVRY_THREADS:
# Number of parallel worker threads during an online restore.
```

```
OFF_RECVRY_THREADS 10              # Default number of offline worker threads
ON_RECVRY_THREADS 1               # Default number of online worker threads

# Data Replication Variables
DRINTERVAL      30              # DR max time between DR buffer flushes (in sec)
DRTIMEOUT       30              # DR network timeout (in sec)
DRAUTO          0               # DR automatic switchover: 0 => Off, 1 => On
DRLOSTFOUND     /usr/informix6/sqldist/etc/dr.lostfound
                                # DR lost+found file path

# Read Ahead Variables
RA_PAGES        4               # Number of pages to attempt to read ahead
RA_THRESHOLD    2               # Number of pages left before next group

# DBSPACETEMP:
# OnLine equivalent of DBTEMP for SE. This is the name of the dbspace
# that the OnLine SQL Engine will use to create temp tables etc.
# The default (if left unspecified) becomes ROOTNAME automatically.
# If specified it must be the name of a dbspace that must exist
# when the OnLine system is brought online otherwise various ad hoc
# queries will return ISAM error 130 (no such dbspace).

DBSPACETEMP                     # Default temp dbspace

# DUMP*:
# The following parameters control the type of diagnostics information which
# is preserved when an unanticipated error condition (assertion failure) occurs
# during OnLine operations.
# For DUMPSHMEM, DUMPGCORE and DUMPCORE 1 means Yes, 0 means No.

DUMPDIR         /tmp            # Preserve diagnostics in this directory
DUMPSHMEM       1               # Dump a copy of shared memory
DUMPGCORE       0               # Dump a core image using 'gcore'
DUMPCORE        0               # Dump a core image (Warning&ml.this abort OnLine)
DUMPCNT         1               # Number of shared memory or gcore dumps for
                     # a single user's session

# ADT*
# The following parameters control the type and level of secure auditing
# present in the OnLine system. By default, ADTMODE is 0 and auditing
# is disabled

ADTMODE         0               # Auditing mode
ADTPATH         /tmp            # Directory where audit trails will be written b
y OnLine
ADTSIZE         50000           # Maximum size of any single audit trail file
ADTERR          0               # 0 ==> retry failed audit writes; 1 ==> log fai
lure

FILLFACTOR      90              # Fill factor for building indexes

# method for OnLine to use when determining current time
USEOSTIME       0               # 0: use internal time(fast), 1: get time from O
S(slow)
```

A.2.2 dbspaces for Database Server Operating on File System Files

```
$ onstat -d

RSAM Version 6.00.UD1   -- On-Line -- Up 00:00:55 -- 9752 Kbytes

Dbspaces
address  number   flags    fchunk   nchunks  flags    owner    name
3000e7c0 1        1        1        1        N        informix rootdbs
3000e804 2        1        2        1        N        informix inf6fs
3000e848 3        1        3        1        N        informix inf6fs2
 3 active, 8 total

Chunks
address  chk/dbs offset size free flags pathname
3000dc0c 1   1   0      5000 933   PO-   /usr/informix6/dbspaces/inf6rootdbs
3000dca4 2   2   0      5000 653   PO-   /usr/informix6/dbspaces/inf6fs
3000dd3c 3   3   0      5000 4947  PO-   /usr/informix6/dbspaces/inf6fs2
 3 active, 8 total
```

A.2.3 File onconfig for Database Server Operating on Raw Devices

```
#**************************************************************************
#
#                  INFORMIX SOFTWARE, INC.
#
#  Title:  onconfig.std
#  Sccsid:      @(#)onconfig.std     9.24 9/30/93    10:04:32
#  Description: INFORMIX-OnLine Configuration Parameters
#
#**************************************************************************

# Root Dbspace Configuration

ROOTNAME        rootdbs         # Root dbspace name
ROOTPATH        /dev/rinf6rootdbs # Path for device containing root dbspace
ROOTOFFSET      0               # Offset of root dbspace into device (Kbytes)
ROOTSIZE        20000           # Size of root dbspace (Kbytes)

# Disk Mirroring Configuration Parameters

MIRROR          0               # Mirroring flag (Yes = 1, No = 0)
MIRRORPATH                      # Path for device containing mirrored root
MIRROROFFSET    0               # Offset into mirrored device (Kbytes)

# Physical Log Configuration

PHYSDBS         rootdbs         # Location (dbspace) of physical log
PHYSFILE        2000            # Physical log file size (Kbytes)

# Logical Log Configuration

LOGFILES        6               # Number of logical log files
LOGSIZE         1000            # Logical log size (Kbytes)

# Message Files
```

```
MSGPATH         /usr/informix6/online.log # System message log file path
CONSOLE         /dev/console    # System console message path

# System Archive Tape Device

TAPEDEV         /home/informix6/tapedev # Tape device path
TAPEBLK         16              # Tape block size (Kbytes)
TAPESIZE        50000           # Maximum amount of data to put on tape (Kbytes)

# Log Archive Tape Device

LTAPEDEV        /home/informix6/ltapedev # Log tape device path
LTAPEBLK        16              # Log tape block size (Kbytes)
LTAPESIZE       10240           # Max amount of data to put on log tape (Kbytes)

# Optical

STAGEBLOB                       # INFORMIX-OnLine/Optical staging area

# System Configuration

SERVERNUM       1               # Unique id corresponding to a OnLine instance
DBSERVERNAME    balticshm       # Name of default database server
DBSERVERALIASES                 # List of alternate dbservernames
DEADLOCK_TIMEOUT 60              # Max time to wait of lock in distributed env.
RESIDENT        0               # Forced residency flag (Yes = 1, No = 0)

MULTIPROCESSOR  0               # 0 for single-processor, 1 for multi-processor
NUMCPUVPS       1               # Number of user (cpu) vps
SINGLE_CPU_VP   0               # If non-zero, limit number of cpu vps to one

NOAGE           0               # Process aging
AFF_SPROC       0               # Affinity start processor
AFF_NPROCS      0               # Affinity number of processors

# Shared Memory Parameters

USERTHREADS     20              # Maximum number of concurrent threads
TRANSACTIONS    20              # Maximum number of concurrent transactions
LOCKS           2000            # Maximum number of locks
BUFFERS         200             # Maximum number of shared buffers
TBLSPACES       200             # Maximum number of open tblspaces
CHUNKS          8               # Maximum number of chunks
NUMAIOVPS       2               # Number of IO vps
DBSPACES        8               # Maximum number of dbspaces
PHYSBUFF        128             # Physical log buffer size (Kbytes)
LOGBUFF         128             # Logical log buffer size (Kbytes)
LOGSMAX         6               # Maximum number of logical log files
CLEANERS        1               # Number of buffer cleaner processes
SHMBASE         0x30000000        # Shared memory base address
SHMVIRTSIZE     8000            # initial virtual shared memory segment size
SHMADD          8192            # Size of new shared memory segments (Kbytes)
SHMTOTAL        0               # Total shared memory (Kbytes). 0=>unlimited
CKPTINTVL       300             # Check point interval (in sec)
LRUS            8               # Number of LRU queues
LRU_MAX_DIRTY   60              # LRU percent dirty begin cleaning limit
LRU_MIN_DIRTY   50              # LRU percent dirty end cleaning limit
LTXHWM          80              # Long transaction high water mark percentage
LTXEHWM         90              # Long transaction high water mark (exclusive)
```

```
TXTIMEOUT       0x12c            # Transaction timeout (in sec)
STACKSIZE       32              # Stack size (Kbytes)

# System Page Size
# BUFFSIZE - OnLine no longer supports this configuration parameter.
#            To determine the page size used by OnLine on your platform
#            see the last line of output from the command, 'onstat -b'.


# Recovery Variables
# OFF_RECVRY_THREADS:
# Number of parallel worker threads during fast recovery or an offline restore.
# ON_RECVRY_THREADS:
# Number of parallel worker threads during an online restore.

OFF_RECVRY_THREADS 10             # Default number of offline worker threads
ON_RECVRY_THREADS 1              # Default number of online worker threads

# Data Replication Variables
DRINTERVAL      30              # DR max time between DR buffer flushes (in sec)
DRTIMEOUT       30              # DR network timeout (in sec)
DRAUTO          0               # DR automatic switchover: 0 => Off, 1 => On
DRLOSTFOUND     /usr/informix6/sqldist/etc/dr.lostfound
                                # DR lost+found file path

# Read Ahead Variables
RA_PAGES        4               # Number of pages to attempt to read ahead
RA_THRESHOLD    2               # Number of pages left before next group

# DBSPACETEMP:
# OnLine equivalent of DBTEMP for SE. This is the name of the dbspace
# that the OnLine SQL Engine will use to create temp tables etc.
# The default (if left unspecified) becomes ROOTNAME automatically.
# If specified it must be the name of a dbspace that must exist
# when the OnLine system is brought online otherwise various ad hoc
# queries will return ISAM error 130 (no such dbspace).

DBSPACETEMP                     # Default temp dbspace

# DUMP*:
# The following parameters control the type of diagnostics information which
# is preserved when an unanticipated error condition (assertion failure) occurs
# during OnLine operations.
# For DUMPSHMEM, DUMPGCORE and DUMPCORE 1 means Yes, 0 means No.

DUMPDIR         /tmp            # Preserve diagnostics in this directory
DUMPSHMEM       1               # Dump a copy of shared memory
DUMPGCORE       0               # Dump a core image using 'gcore'
DUMPCORE        0               # Dump a core image (Warning:this aborts OnLine)
DUMPCNT         1               # Number of shared memory or gcore dumps for
                     # a single user's session

# ADT*
# The following parameters control the type and level of secure auditing
# present in the OnLine system. By default, ADTMODE is 0 and auditing
# is disabled

ADTMODE         0               # Auditing mode
ADTPATH         /tmp            # Directory where audit trails will be written b
```

```
y OnLine
ADTSIZE         50000           # Maximum size of any single audit trail file
ADTERR          0               # 0 ==> retry failed audit writes; 1 ==> log fai
lure

FILLFACTOR      90              # Fill factor for building indexes

# method for OnLine to use when determining current time
USEOSTIME       0               # 0: use internal time(fast), 1: get time from O
S(slow)
```

A.2.4 dbspaces for Database Server Operating on Raw Devices

```
$ onstat -d

RSAM Version 6.00.UD1   -- On-Line -- Up 00:01:54 -- 9752 Kbytes

Dbspaces
address  number   flags    fchunk   nchunks  flags    owner    name
3000e200 1        1        1        1        N        informix rootdbs
3000e244 2        4        2        1        N        informix inf6rd
 2 active, 8 total

Chunks
address  chk/dbs offset   size     free     bpages   flags pathname
3000dc0c 1   1   0        5000     2331              PO-   /dev/rinf6rootdbs
3000dca4 2   2   0        5000     1755              PO-   /dev/rinf6rd
 2 active, 8 total
```

A.2.5 File config.arc for the Onarchive Utility

```
!*****************************************************************************
!*
!*                 INFORMIX SOFTWARE, INC.
!*
!*                    PROPRIETARY DATA
!*
!*   THIS DOCUMENT CONTAINS TRADE SECRET DATA WHICH IS THE PROPERTY OF
!*   INFORMIX SOFTWARE, INC.  THIS DOCUMENT IS SUBMITTED TO RECIPIENT IN
!*   CONFIDENCE.  INFORMATION CONTAINED HEREIN MAY NOT BE USED, COPIED OR
!*   DISCLOSED IN WHOLE OR IN PART EXCEPT AS PERMITTED BY WRITTEN AGREEMENT
!*   SIGNED BY AN OFFICER OF INFORMIX SOFTWARE, INC.
!*
!*   THIS MATERIAL IS ALSO COPYRIGHTED AS AN UNPUBLISHED WORK UNDER
!*   SECTIONS 104 AND 408 OF TITLE 17 OF THE UNITED STATES CODE.
!*   UNAUTHORIZED USE, COPYING OR OTHER REPRODUCTION IS PROHIBITED BY LAW.
!*
!*
!*  Sccsid:  @(#)config.arc     9.11 8/16/93  15:43:30
!*
!*****************************************************************************
!*/

! First line must specify which language is used in CONFIG file.
! This language is used as starting language for onarchive.
```

```
! It can be changed in the operator default file and/or in the
! user default file.

ENGLISH

! Device available for onarchive.
DEVICE HO          = /home
DEVICE ho          = /home
DEVICE DISK        = /home/informix6
DEVICE disk        = /home/informix6
DEVICE TAPE        = /home/informix6/tapedev
DEVICE tape        = /home/informix6/tapedev
DEVICE TAPE_VOP    = /home/informix6/tapedev
DEVICE tape_vop    = /home/informix6/tapedev

! NOTE: all the following file names are relative to INFORMIXDIR
! Operator default file.
DEFAULT         = /etc/oper_deflt.arc


! Error messages files.
ERROR ENGLISH   = /msg/errmsg_E.dat

! Format files.
MESSAGE ENGLISH = /msg/fmt_E.dat

! Help files.
HELP ENGLISH    = /msg/help_E.hpf

KEYM_HELP ENGLISH = /msg/hlp_keym_E.hpf

! Catalog message file.
CATALOG MESSAGE = /msg/caterr_E.dat

! User privileges
PRIVILEGE = OPERATOR
!PRIVILEGE = OPERATOR, GROUP, OWNER

! Timeout value (in minutes)
TIME_OUT = 30

! Number of buffers allocated to write on DISK
! The size of a buffer is equal to /BLOCKSIZE used by Archive and Copy commands.
NB_DISK_SPACE_EXTENT = 10
```

A.2.6 File oper_deflt.arc for the Onarchive Utility

```
!*****************************************************************************
!*
!*                 INFORMIX SOFTWARE, INC.
!*
!*                    PROPRIETARY DATA
!*
!*   THIS DOCUMENT CONTAINS TRADE SECRET DATA WHICH IS THE PROPERTY OF
!*   INFORMIX SOFTWARE, INC.  THIS DOCUMENT IS SUBMITTED TO RECIPIENT IN
!*   CONFIDENCE.  INFORMATION CONTAINED HEREIN MAY NOT BE USED, COPIED OR
```

```
!*   DISCLOSED IN WHOLE OR IN PART EXCEPT AS PERMITTED BY WRITTEN AGREEMENT
!*   SIGNED BY AN OFFICER OF INFORMIX SOFTWARE, INC.
!*
!*   THIS MATERIAL IS ALSO COPYRIGHTED AS AN UNPUBLISHED WORK UNDER
!*   SECTIONS 104 AND 408 OF TITLE 17 OF THE UNITED STATES CODE.
!*   UNAUTHORIZED USE, COPYING OR OTHER REPRODUCTION IS PROHIBITED BY LAW.
!*
!*
!*  Sccsid:  @(#)oper_deflt.arc      9.9 10/4/93  16:17:07
!*
!*****************************************************************************
!*/
! This file is used only if the users haven't defined their own default file.

! Starting language (override the configuration language).
ENGLISH

! These defaults are the same as the system's defaults.
! Modified as needed.
/NOBAART
/BLOCKSIZE=65536
/BRIEF
/COMPRESS=REP
/NOCRC
/DECOMPRESS=REP
/COPIES=1
/NODECRYPT
/DENSITY=0
/NOEXPIRY_DATE
/NOENCRYPT
/LEVEL=0
/NOLOG
/MAX_SPACE=0
/NONOTIFY
/PROTECTION=RWD
/NOTRANSIT
/NOVERIFY
```

A.3 INFORMIX-OnLine Version 7.21 Configuration Files

Here are examples of the onconfig, sqlhosts, oncfg, and emergency boot files used in our test environment.

A.3.1 onconfig File

```
#*****************************************************************************
#
#                  INFORMIX SOFTWARE, INC.
#
#  Title:  onconfig.std
#  Description: INFORMIX-OnLine Configuration Parameters
#
#*****************************************************************************

# Root Dbspace Configuration

ROOTNAME        rootdbs         # Root dbspace name
ROOTPATH        /dev/informix/learn # Path for device containing root dbspace
```

```
ROOTOFFSET      4               # Offset of root dbspace into device (Kbytes)
ROOTSIZE        20000           # Size of root dbspace (Kbytes)

# Disk Mirroring Configuration Parameters

MIRROR          1               # Mirroring flag (Yes = 1, No = 0)
MIRRORPATH                      # Path for device containing mirrored root
MIRROROFFSET    4               # Offset into mirrored device (Kbytes)

# Physical Log Configuration

PHYSDBS         rootdbs         # Location (dbspace) of physical log
PHYSFILE        2000            # Physical log file size (Kbytes)

# Logical Log Configuration

LOGFILES        6               # Number of logical log files
LOGSIZE         1000            # Logical log size (Kbytes)

# Diagnostics

MSGPATH         /usr/informix/7.2/learn.log # System message log file path
CONSOLE         /dev/console    # System console message path
ALARMPROGRAM    /usr/informix/7.2/etc/log_full.sh # Alarm program path

# System Archive Tape Device

TAPEDEV         /tmp/tapedev    # Tape device path
TAPEBLK         16              # Tape block size (Kbytes)
TAPESIZE        10240           # Maximum amount of data to put on tape (Kbytes)

# Log Archive Tape Device

LTAPEDEV        /tmp/tapedev    # Log tape device path
LTAPEBLK        16              # Log tape block size (Kbytes)
LTAPESIZE       10240           # Max amount of data to put on log tape (Kbytes)

# Optical

STAGEBLOB                       # INFORMIX-OnLine/Optical staging area

# System Configuration

SERVERNUM       1               # Unique id corresponding to a OnLine instance
DBSERVERNAME    learn           # Name of default database server
DBSERVERALIASES learn_online    # List of alternate dbservernames
DEADLOCK_TIMEOUT 60              # Max time to wait of lock in distributed env.
RESIDENT        0               # Forced residency flag (Yes = 1, No = 0)

MULTIPROCESSOR  0               # 0 for single-processor, 1 for multi-processor
NUMCPUVPS       1               # Number of user (cpu) vps
SINGLE_CPU_VP   0               # If non-zero, limit number of cpu vps to one

NOAGE           0               # Process aging
AFF_SPROC       0               # Affinity start processor
AFF_NPROCS      0               # Affinity number of processors

# Shared Memory Parameters
```

```
LOCKS           2000            # Maximum number of locks
BUFFERS         200             # Maximum number of shared buffers
NUMAIOVPS                       # Number of IO vps
PHYSBUFF        32              # Physical log buffer size (Kbytes)
LOGBUFF         32              # Logical log buffer size (Kbytes)
LOGSMAX         50              # Maximum number of logical log files
CLEANERS        1               # Number of buffer cleaner processes
SHMBASE         0x30000000        # Shared memory base address
SHMVIRTSIZE     8000            # initial virtual shared memory segment size
SHMADD          8192            # Size of new shared memory segments (Kbytes)
SHMTOTAL        0               # Total shared memory (Kbytes). 0=>unlimited
CKPTINTVL       300             # Check point interval (in sec)
LRUS            8               # Number of LRU queues
LRU_MAX_DIRTY   60              # LRU percent dirty begin cleaning limit
LRU_MIN_DIRTY   50              # LRU percent dirty end cleaning limit
LTXHWM          50              # Long transaction high water mark percentage
LTXEHWM         60              # Long transaction high water mark (exclusive)
TXTIMEOUT       0x12c             # Transaction timeout (in sec)
STACKSIZE       32              # Stack size (Kbytes)

# System Page Size
# BUFFSIZE - OnLine no longer supports this configuration parameter.
#            To determine the page size used by OnLine on your platform
#            see the last line of output from the command, 'onstat -b'.


# Recovery Variables
# OFF_RECVRY_THREADS:
# Number of parallel worker threads during fast recovery or an offline restore.
# ON_RECVRY_THREADS:
# Number of parallel worker threads during an online restore.

OFF_RECVRY_THREADS 10             # Default number of offline worker threads
ON_RECVRY_THREADS 1              # Default number of online worker threads

# Data Replication Variables
# DRAUTO: 0 manual, 1 retain type, 2 reverse type
DRAUTO          0               # DR automatic switchover
DRINTERVAL      30              # DR max time between DR buffer flushes (in sec)
DRTIMEOUT       30              # DR network timeout (in sec)
DRLOSTFOUND     /usr/informix/etc/dr.lostfound # DR lost+found file path

# Backup/Restore variables
BAR_ACT_LOG     /tmp/bar_act.log
BAR_DEBUG       9
BAR_MAX_BACKUP  0
BAR_RETRY       0
BAR_NB_XPORT_COUNT 40
BAR_XFER_BUF_SIZE 4


# Read Ahead Variables
RA_PAGES                        # Number of pages to attempt to read ahead
RA_THRESHOLD                    # Number of pages left before next group

# DBSPACETEMP:
# OnLine equivalent of DBTEMP for SE. This is the list of dbspaces
# that the OnLine SQL Engine will use to create temp tables etc.
# If specified it must be a colon separated list of dbspaces that exist
```

```
# when the OnLine system is brought online.  If not specified, or if
# all dbspaces specified are invalid, various ad hoc queries will create
# temporary files in /tmp instead.

DBSPACETEMP                     # Default temp dbspaces

# DUMP*:
# The following parameters control the type of diagnostics information which
# is preserved when an unanticipated error condition (assertion failure) occurs
# during OnLine operations.
# For DUMPSHMEM, DUMPGCORE and DUMPCORE 1 means Yes, 0 means No.

DUMPDIR         /tmp            # Preserve diagnostics in this directory
DUMPSHMEM       1               # Dump a copy of shared memory
DUMPGCORE       0               # Dump a core image using 'gcore'
DUMPCORE        0               # Dump a core image (Warning:this aborts OnLine)
DUMPCNT         1               # Number of shared memory or gcore dumps for
                     # a single user's session

FILLFACTOR      90              # Fill factor for building indexes

# method for OnLine to use when determining current time
USEOSTIME       0               # 0: use internal time(fast), 1: get time from OS(slow)

# Parallel Database Queries (pdq)
MAX_PDQPRIORITY 100             # Maximum allowed pdqpriority
DS_MAX_QUERIES                  # Maximum number of decision support queries
DS_TOTAL_MEMORY                 # Decision support memory (Kbytes)
DS_MAX_SCANS    1048576         # Maximum number of decision support scans
DATASKIP        off             # List of dbspaces to skip

# OPTCOMPIND
# 0 => Nested loop joins will be preferred (where
#      possible) over sortmerge joins and hash joins.
# 1 => If the transaction isolation mode is not
#      "repeatable read", optimizer behaves as in (2)
#      below.  Otherwise it behaves as in (0) above.
# 2 => Use costs regardless of the transaction isolation
#      mode.  Nested loop joins are not necessarily
#      preferred.  Optimizer bases its decision purely
#      on costs.
OPTCOMPIND      2               # To hint the optimizer

ONDBSPACEDOWN   0               # Dbspace down option: 0 = CONTINUE, 1 = ABORT, 2 = WAIT
LBU_PRESERVE    0               # Preserve last log for log backup
OPCACHEMAX      0               # Maximum optical cache size (Kbytes)

# HETERO_COMMIT (Gateway participation in distributed transactions)
# 1 => Heterogeneous Commit is enabled
# 0 (or any other value) => Heterogeneous Commit is disabled
HETERO_COMMIT   0
```

A.3.2 sqlhost File

```
#************************************************************************
#
#                  INFORMIX SOFTWARE, INC.
#
#                     PROPRIETARY DATA
#
#    THIS DOCUMENT CONTAINS TRADE SECRET DATA WHICH IS THE PROPERTY OF
#    INFORMIX SOFTWARE, INC.  THIS DOCUMENT IS SUBMITTED TO RECIPIENT IN
#    CONFIDENCE.  INFORMATION CONTAINED HEREIN MAY NOT BE USED, COPIED OR
#    DISCLOSED IN WHOLE OR IN PART EXCEPT AS PERMITTED BY WRITTEN AGREEMENT
#    SIGNED BY AN OFFICER OF INFORMIX SOFTWARE, INC.
#
#    THIS MATERIAL IS ALSO COPYRIGHTED AS AN UNPUBLISHED WORK UNDER
#    SECTIONS 104 AND 408 OF TITLE 17 OF THE UNITED STATES CODE.
#    UNAUTHORIZED USE, COPYING OR OTHER REPRODUCTION IS PROHIBITED BY LAW.
#
#
#   Title:      sqlhosts.demo
#   Sccsid:     @(#)sqlhosts.demo    9.2  7/15/93  15:20:45
#   Description:
#         Default sqlhosts file for running demos.
#
#************************************************************************

learn onipcshm  bering          learn
learn_online onsoctcp     bering          xyz
```

A.3.3 oncfg File

```
DBSpace 1 1 1 1 846814278 14 0 4 926112 4 -1 1073932088 rootdbs            informix
DBSpace 2 1 2 1 846871800 4 0 0 0 0 0 1077399160 happyspace         informix
Chunk 1 0 1 5000 1985 1 0 64 19 /dev/informix/learn
Chunk 2 0 1 4000 3947 2 0 64 22 /dev/informix/dbspace1
Log 1 0 6423 846814323 0 1049139 250 0
Log 2 0 11264 846814340 0 1049389 250 0
Log 3 0 16187 846814365 0 1049639 250 0
Log 4 19 0 0 4 1049889 250 240
Log 5 0 0 0 0 1050139 250 0
Log 6 0 0 0 0 1050389 250 0
```

A.3.4 Emergency Boot File

```
 learn rootdbs R 1 1 0 0 63489 1996-10-31 18:18:01 1
 learn rootdbs R 0 2 1 0 63490 1996-10-31 18:19:12 1
 learn rootdbs R 0 3 1 0 63491 1996-10-31 18:23:26 1
```

A.4 SYBASE V4.10 Configuration

A.4.1 SQL Server Configuration

name	minimum	maximum	config_value	run_value
recovery interval	1	32767	0	5
allow updates	0	1	1	1
user connections	5	2147483647	0	25
memory	3850	2147483647	0	5120
open databases	5	2147483647	0	10
locks	5000	2147483647	0	5000
open objects	100	2147483647	0	500
procedure cache	1	99	0	20
fill factor	0	100	0	0
time slice	50	1000	0	100
database size	2	10000	0	2
tape retention	0	365	0	0
recovery flags	0	1	0	0
nested triggers	0	1	1	1
devices	4	256	0	10
remote access	0	1	1	1
remote logins	0	2147483647	0	20
remote sites	0	2147483647	0	10
remote connections	0	2147483647	0	20
pre-read packets	0	2147483647	0	3
upgrade version	0	2147483647	1000	1000
default sortorder id	0	255	50	50
default language	0	2147483647	0	0
language in cache	3	100	3	3
max online engines	1	32	1	1
min online engines	1	32	1	1
engine adjust interval	1	32	0	0
cpu flush	1	2147483647	200	200
i/o flush	1	2147483647	1000	1000
default character set id	0	255	1	1
stack size	20480	2147483647	0	28672
password expiration interval	0	32767	0	0
audit queue size	1	65535	100	100
additional netmem	0	2147483647	0	0
default network packet size	512	524288	0	512
maximum network packet size	512	524288	0	512
extent i/o buffers	0	2147483647	0	0
identity burning set factor	1	9999999	5000	5000

A.4.2 Database Configurations

The configuration information is for the Sybase database installed on file system files, **DBTEST**, and the Sybase database installed on raw devices, **RAWDB**.

DBTEST Database

name	db_size	owner	dbid
dbtest	6.0 MB	sa	8

device_fragments	size	usage	free kbytes
dev1	2.0 MB	data only	1280
dev2	2.0 MB	log only	2032
log_dev	2.0 MB	data only	2048

RAWDB Database

```
name        db_size      owner        dbid ------------  --------    ----------     -------
rawdb       4.0 MB       sa           9

device_fragments            size        usage             free kbytes
------------------------------ ------------- -------------------- -----------
raw_disk                    4.0 MB       data and log             3408
```

A.5 DB2/6000 V1.1 Configuration

A.5.1 Database Manager Configuration

Database Manager Configuration

```
Database manager configuration release level  = 0x0500
Node type                                      = Server with remote clien

Service name                                (SVCENAME) =
Transaction program name                      (TPNAME) =

Max requester I/O block size (bytes)         (RQRIOBLK) = 4096
Max server I/O block size (bytes)            (SVRIOBLK) = 4096
Communication heap size (4KB)               (COMHEAPSZ) = 128
Remote services heap size (4KB)              (RSHEAPSZ) = 128
Sort heap threshold (4KB)                   (SHEAPTHRES) = 4096
Application support layer heap size (4KB)    (ASLHEAPSZ) = 100

Max no. of existing agents                   (MAXAGENTS) = 200
Max no. of concurrent agents                (MAXCAGENTS) = MAXAGENTS
Max no. of concurrently active databases          (NUMDB) = 8
Application cleanup interval (ms)           (CUINTERVAL) = 5000
Keep DARI process                             (KEEPDARI) = YES
Max. no. of DARI processes                     (MAXDARI) = MAXAGENTS
Priority of agents                            (AGENTPRI) = SYSTEM
Database monitor SQL statement size (bytes) (SQLSTMTSZ) = 256
Index re-creation time                         (INDEXREC) = RESTART
Default database path                         (DFTDBPATH) = /home/db2

Backup buffer default size (4KB)             (BACKBUFSZ) = 1024
Restore buffer default size (4KB)            (RESTBUFSZ) = 1024
```

A.5.2 Database Configuration

Database Configuration for Database ibmsampl

```
Max no. of active applications               (MAXAPPLS) = 20
Max DB files open per appl.                  (MAXFILOP) = 64
Default application heap (4KB)              (APPLHEAPSZ) = 48
Package cache size (4KB)                    (PCKCACHESZ) = 4

Max storage for lock lists (4KB)             (LOCKLIST) = 63
Percent. of lock lists per appl.             (MAXLOCKS) = 10
Interval for checking deadlock (ms)         (DLCHKTIME) = 10000

Buffer pool size (4KB)                       (BUFFPAGE) = 1000
```

```
 Database heap (4KB)                          (DBHEAP) = 128
 SQL statement heap (4KB)                    (STMTHEAP) = 1024
 Sort list heap (4KB)                        (SORTHEAP) = 256

 Log buffer size                             (LOGBUFSZ) = 8
 %  log file reclaimed before soft checkpoint (SOFTMAX) = 100
 Group commit count                         (MINCOMMIT) = 1
 Log file size (4KB)                        (LOGFILSIZ) = 100
 No. of primary log files                  (LOGPRIMARY) = 10
 No. of secondary log files                 (LOGSECOND) = 2
 Path to log files                                      = /home/db2/db2/SQL00001
/SQLOGDIR/
 Changed path to log files                (NEWLOGPATH) =
 First active log file                                  = S0000005.LOG
 Next active log file                                   = S0000005.LOG

 Database attributes                           (DBATTR) = 14
   Log retain for recovery enabled           (LOGRETAIN) = ON
   User exit for logging enabled              (USEREXIT) = ON
   Auto restart enabled                     (AUTORESTART) = ON

 Index re-creation time                        (INDEXREC) = SYSTEM (RESTART)

 Number of segments                                     = 32
 Maximum file segment size (4KB)                        = 76800

 Database State                                         = 49
   Database is consistent                               = YES
   Backup pending                                       = NO
   Rollforward pending                                  = NO
   Log retain for recovery status                       = ON
   User exit for logging status                         = ON

 Database territory                                     = En_US
 Database code set                                      = IBM-850
 Database country code                                  = 1
 Database code page                                     = 850
 Database configuration release level                   = 0x0500
```

A.6 DB2/2 V1.2 Configuration

A.6.1 Database Manager Configuration

Database Manager Configuration

```
 Max requester I/O block size              (RQRIOBLK) = 4096
 Max server I/O block size                 (SVRIOBLK) = 0
 Max kernel segments                       (SQLENSEG) = 25
 Max no. of active databases                  (NUMDB) = 3
 Workstation name                             (NNAME) = PERSIAN
 Communication heap size                  (COMHEAPSZ) = 2
 Remote services heap size                 (RSHEAPSZ) = 2
 Max remote connections                       (NUMRC) = 3
 Sort heap threshold                     (SHEAPTHRES) = 250

 Index recreation time                     (INDEXREC) = ACCESS
```

```
Workstation type                     = Client with local databases
Database Manager release level                 = 0x0400
```

A.6.2 Database Configuration

Database Configuration for Database SAMPLE

```
Max. no. of active applications          (MAXAPPLS) = 8
Max. DB files open per appl.             (MAXFILOP) = 20
Max. files open per appl.             (MAXTOTFILOP) = 255
Default application heap                (APPLHEAPSZ) = 3
Application agent heap                   (AGENTHEAP) = 2

Max. storage for lock list               (LOCKLIST) = 8
Percent of lock list allowed             (MAXLOCKS) = 22
Deadlock check time interval            (DLCHKTIME) = 10000

Buffer pool size                         (BUFFPAGE) = 25
Database heap                              (DBHEAP) = 2
SQL statement heap                       (STMTHEAP) = 64
Sort list heap                           (SORTHEAP) = 2

No. log records for checkpoint            (SOFTMAX) = 100
Log file size                           (LOGFILSIZ) = 32
No. of primary log files                (LOGPRIMARY) = 3
No. of secondary log files               (LOGSECOND) = 2
Path to log file                                    = D:\SQL00001\SQLOGDIR\
Changed path to log file                (NEWLOGPATH) =
First active log file                               = S0000011.LOG
Next active log file                                = S0000014.LOG

Database attributes                          (DBATTR) = 15
  Copy protect enable                  (COPYPROTECT) = ON
  Log retain enable                      (LOGRETAIN) = ON
  User exit enable                       (USEREXIT) = ON
  Auto restart enable                  (AUTORESTART) = ON

Index recreation time                     (INDEXREC) = SYSTEM (ACCESS)

Database State                                      = 48
  Database is consistent                            = NO
  Backup pending                                    = NO
  Rollforward pending                               = NO
  Log retain status                                 = ON
  User exit status                                  = ON

Database country code                               = 1
Database code page                                  = 437
Database release level                              = 0x0400
```

Appendix B. AIX and OS/2 Backup Utilities

This appendix provides details on the backup utilities that come with the AIX and OS/2 operating systems. First we discuss the difference between using the AIX utilities to back up files by either their name or inode. Then we look the AIX backup, restore, tar, dd, mksysb, and cpio utilities. We conclude with a look at the OS/2 backup, xcopy, and pkzip2 utilities.

B.1 AIX Backup Utilities

B.1.1 Backup by File Name and inode

In an AIX environment it is possible to back up files by either their file name or inode information. Each AIX utility provides one or both options.

A backup by file name stores files with either their relative or absolute file names. There is more flexibility in the restore process with relative names. You can restore files to another directory because they have been backed up relative to their directory. If you use absolute path names when performing the backup, you can only restore files to the same directory from which they were backed up. In most cases, when you restore files, any directories will be created automatically.

A backup by inode is a complete backup of an entire file system. You can only back up and restore files with absolute file names, because the inode information of each file is absolute to the control block (superblock) of the file system. Although a file is represented by its inode information, it can be easily restored by name, but only with its absolute file name in the same file system where it was before.

B.1.2 AIX Backup and Restore Commands

The AIX backup command provides:

- Incremental or nonincremental backup
- Absolute or relative path names
- Backup by file name or inode.

The results of the backup command can be read only by the restore command. The backup and restore commands are AIX only. They are not standard UNIX commands. Here are some options for and examples of this command. For a detailed description, see the *AIX Commands Reference* or the online help facility, *InfoExplorer**.

Useful options (for backup by file name):

-b blocksize	Specifies in kilobytes the block size of the tape device.
-f device	Names the path name of the tape device or file where you want the ASCII data files stored.
-i	Prompts for interactive mode or backup by file name.
-p	Packs the data on the device.
-v	Verifies the backup operation.

The following commands back up the entire /home file system with relative path names to /home on the /dev/rmt0 tape device:

```
cd /home
find . -print | backup -ivf /dev/rmt0
```

Before you issue the backup command to perform a backup with relative path names, you must change your directory to the directory containing the files you are to back up, /home in this case. Files contained in /home will be backed up relatively. If you restore the /home directory, you should be in the /home directory when issuing the restore command. If you were, for instance, in the /tmp directory, the data would then be restored in the /tmp directory.

The following command backs up the entire /home file system with absolute path names by file name on the /dev/rmt0 tape device:

```
find /home -print | backup -ivf /dev/rmt0
```

The restore command restores data that was stored with the backup command.

Useful options (for restore by file name):

-b blocksize	Specifies in kilobytes the block size of the tape device.
-f device	Names the path name of the tape device or file from where you want the ASCII data files to be restored.
-x	Extracts the files from the media by name.
-v	Verifies the backup operation.
-T	Lists a table of contents.

The following command lists a table of contents of the data on the /dev/rmt0 tape device:

```
restore -tvf /dev/rmt0
```

The following commands restore the entire /home file system with the relative path name to the /home directory:

```
cd /home
restore -xvf /dev/rmt0
```

The following commands restore only the file called *georg* with the relative path name to the /home directory:

```
cd /home
  restore -xvf /dev/rmt0 ./georg
```

Note: You must put the ./ in front of the file name or the full relative path of the file that was backed up. For example, *./mydir/georg*.

The following command restores the entire data contents of the tape from the /dev/rmt0 tape device, either with relative or absolute path names:

```
restore -xvf /dev/rmt0
```

B.1.3 AIX tar Command

The AIX tar (tape archiver) command stores files with either absolute or relative path names. The tar command also archives and restores files in non-AIX operating system environments. You would use the tar command instead of the backup command when you want to share data with non-AIX systems because most UNIX operating systems can read the output of the tar command.

Useful options:

-b blocks	Specifies the tape block size.
-c	Writes the data to the archiving device.
-f device	Names the path name of the tape device or file where you want the ASCII data files stored.
-L inputList	Specifies the list of file names to be processed by the tar command.
-t	Lists a table of contents.
-v	Verifies the tar operation.
-x	Extracts files from the archiving device.

The following commands archive the entire /home file system with relative path names on the /dev/rmt0 tape device:

```
cd /
  tar -cvf /dev/rmt0 ./home
```

If you do not want to archive all files of a directory or a file system, you can use the find command in combination with the tar command. First search for specific files with the find command and the -name parameter and direct these selected files to another file. The tar command then reads this file and archives only the files contained in the list. The following commands archive only the files beginning with the prefix *inf* of the /home file system with relative path names on the /dev/rmt0 tape device:

```
  cd /home
  find . -name inf*.* -print > tarlist
  tar -cvf /dev/rmt0 -L tarlist
```

The find command creates a list of files to be archived by the tar command and writes the contents of the list to a file called tarlist. The tar command uses this tarlist file with the -L option to archive the selected files.

The following command archives the entire /home file system with absolute path names on the /dev/rmt0 tape device:

```
  tar -cvf/dev/rmt0 /home
```

B.1.4 AIX dd Command

The AIX dd (device to device copy) command is a special filter command. It is used to make backup copies that are exact images of the file systems. It converts, copies, or filters data that comes from an input file and directs the filtered information to an output file or device. If you want to back up AIX file system files, it is better to use the AIX tar or backup commands. However, if you want to convert data from raw devices to file system files, the dd command performs the data conversion and rebuilds the internal structure of the raw device on the newly created file system file.

Using the AIX dd command in an RDBMS environment is only one approach to converting the data of raw devices into a file system file. It does not guarantee that the converted data contains all information needed to rebuild the database, because the dd command performs the conversion of data at a lower level than the database's point of view.

Useful options:

if=InFile Specifies the input file name from where the data is received.

of=OutFile Specifies the output file name where the data is to be copied.

The following command reads the incoming data from the special device file called /dev/rinf5rd (which is the device-specific file name for the inf5rd raw device) and copies the data to a file system file called /usr/informix5/dbspaces/inf5fs.dd:

```
dd if=/dev/rinf5rd of=/usr/informix5/dbspaces/inf5fs.dd
```

B.1.5 AIX mksysb (make system backup) Command

One of the most frequently used commands when performing backup activities is the AIX mksysb command. This command creates a full system backup of all mounted file systems in the root volume group, rootvg. The mksysb command does not back up unmounted file systems (even though they exist on the disk) or volume groups other than rootvg. After a system crash or one or more disk failures, it may be necessary to reinstall the operating system. This would be done with a tape made by the mksysb command. The mksysb command stores all files of the root volume group as well as the information of the Object Data Manager (ODM) database. This information is needed to re-create the defined devices, such as terminal and printer definitions, logical and physical volume layouts, communication protocol information, and user definitions.

Because the AIX mksysb command in AIX Version 3.2.5 uses the AIX tar command to back up all mounted file system files of the root volume group, rootvg, it does not back up the data contents of raw devices that the RDBMS is using for its data.

The full system backup can be performed through either the AIX System Management Interface Tool (SMIT) or the command line interface. The command line interface is useful if you are performing a scheduled offline system backup with the UNIX cron facility or using the system backup feature in a shell script.

Figure 372 on page 549 shows an example of using the SMIT mksysb interface.

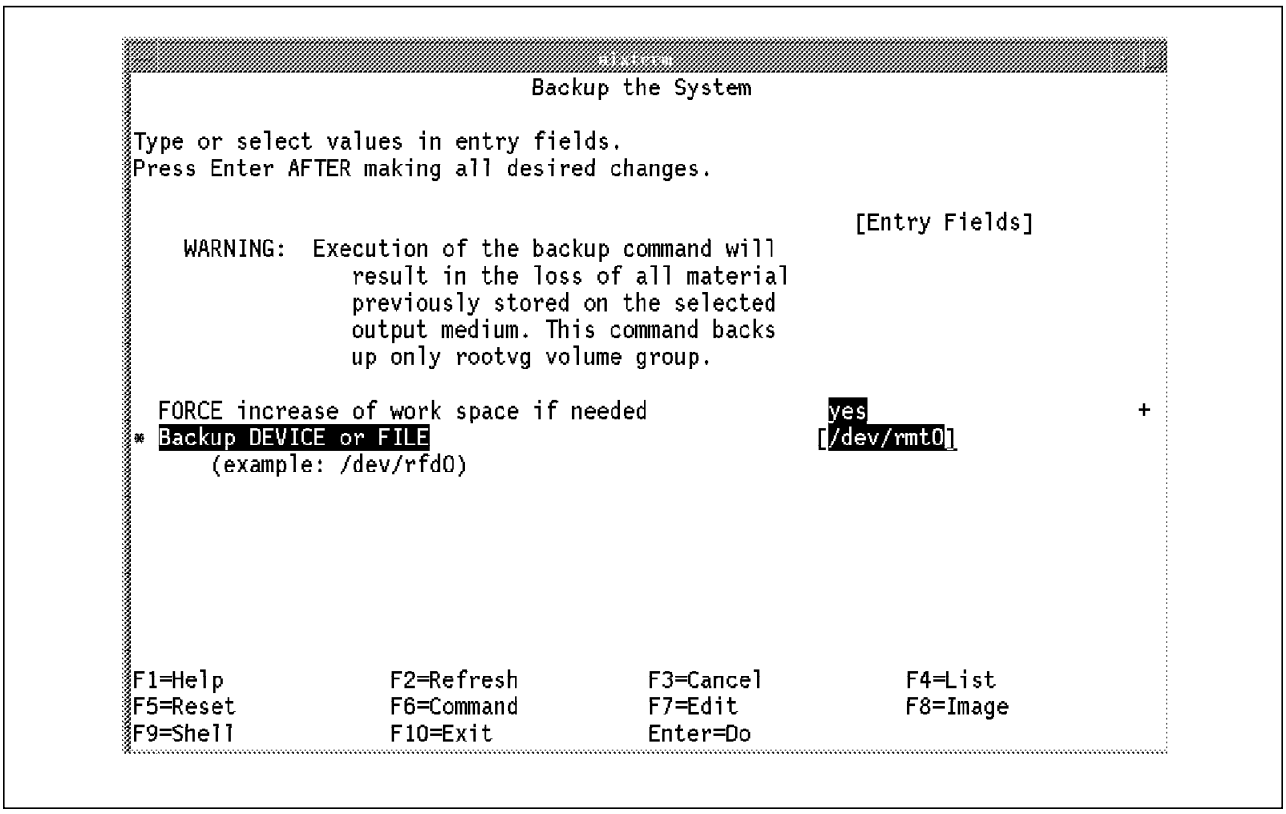


Figure 372. Using mksysb to Back Up the System SMIT Window

When performing the system backup through the command line interface, you first issue an mkszfile command. This mkszfile command determines the file system sizes of all mounted file systems in the root volume group. The output of the mkszfile command is directed to the input of the mksysb command. Within the mkszfile command you should always use the -f option because it forces the increase of work space if needed. If you omit the -f option, creation of the bootable system backup tape with the mksysb command could fail because there would not be enough work space.

The following example creates a full system backup on the /dev/rmt0 tape device:

```
mkszfile -f
mksysb /dev/rmt0
```

You can use the mkszfile and mksysb commands together, connected by an ampersand (&), or you can perform the commands separately.

B.1.6 AIX cpio Command

The AIX cpio (copy input to output) command is used for copying individual files. The files are saved on the backup medium (diskette, disk, or tape) in a special cpio format. Pattern matching with wildcards is provided as well as traversing subdirectories. Date or time criteria are not provided. Also, files with uids and gids greater than 65535 cannot be backed up with cpio; instead use the backup and restore commands.

The following example copies all files with the path names listed in the file names file (in a compact form) onto the diskette (>/dev/rfd0):

```
cpio -ov <filenames >/dev/rfd0
```

The v flag causes a display of each file as it is copied.

The following example copies the current directory and all subdirectories onto diskette. The -print option displays the names of each file as it is copied.

```
find . -cpio /dev/rfd0 -print
```

B.2 OS/2 Backup Utilities

Now let us look at OS/2 backup utilities. The **backup** and **xcopy** commands come native with the OS/2 operating system. The **pkzip2** tool is a PKWARE product that is widely used within IBM.

B.2.1 OS/2 backup Command

The backup command provides basic backup functions to back up one or more files from your hard drive to a floppy disk. Selection criteria are provided to back up files since a date or time specified, back up only files that have changed since the last backup, and back up subdirectories. Extended attributes of the files and directories are also backed up. An extended attribute is a special area used for storing data that describes the file to OS/2 or an application. The backup uses a special format for the data, so you must use the OS/2 restore command to restore the files.

The backup command creates two files in the root directory of the target diskette, BACKUP.XXX, which contains all of the backed up files, and CONTROL.XXX, which contains the path names, file names, and other control information. You also have the option to create a BACKUP.LOG file, which contains the when, where, and what for the backup.

Backup will not handle system files (command.com, cmd.exe, os2.ini, os2sys.ini), hidden system files (wp root.sf, ea data.sf), and open dynamic link library (DLL) files. You can use xcopy to back up those files.

The following example backs up all files on the C drive's root directory onto the A drive, including files in subdirectories, and creates a log file called BACKUP.LOG:

```
Backup C: A: /S /L
```

The following example backs up all files in the C drive's root directory that have been changed since August 1, 1994 at 9 a.m.:

```
Backup C: A: /D:08-01-94 /T:09:00:00
```

B.2.2 OS/2 xcopy Command

The xcopy command, or extended copy, is a specialized version of the copy command. It handles groups of files more flexibly. Although it is really a copy and not a backup command, it can be used for backup and, in some cases, handle files that the backup command cannot (for example, the os2.ini and os2sys.ini files).

Xcopy copies groups of files or subdirectories and is a convenient way to copy large numbers of files. You can control which files are copied by using selection

criteria such as the date, location, and archive attribute. Extended attributes are copied if you so specify.

The following example copies all files and subdirectories on the A drive to the C drive, even the empty directories:

```
xcopy A:\ C:\ /S /E
```

The following example copies every file in the \memos directory with a date later than 1991:

```
xcopy \memos A:\ /D:1-1-92
```

B.2.3 OS/2 pkzip2 Tool

The OS/2 pkzip2 tool is a shareware package from PKWARE, Inc. that provides both compression and archiving functions. PKZIP2 and PKUNZIP2 are the protected mode versions of PKZIP and PKUNZIP. Protected mode means that the tool will run only in an OS/2 window. The family mode versions (PKZIPF and PKUNZIPF) run in either OS/2 or DOS windows.

With PKZIP you can pick multiple files to archive or compress and you can use wildcards to select the files. Options exist to add changed files to a pkzip file. You can also optionally traverse subdirectories, choose files based on a specified date, and include hidden, system, or read-only files.

Appendix C. ADSMPIPE: Raw Logical Volume Backups with the API

This appendix describes a package we updated. It was originally written by Tim Bell from IBM High Energy Physics support, IBM Switzerland. This package, **ADSMPIPE**, is a straightforward utilization of the ADSM API that provides archive, backup, retrieve, and restore facilities for any data that can be piped into it. It has been modified to add a password change facility. As discussed in 4.2, "Techniques for Using ADSM to Back Up Databases" on page 51, ADSMPIPE is provided as an "as-is" utility and is not supported.

In the past it was not possible for ADSM to directly read a raw logical volume. You could use an RDBMS backup utility (which used the ADSM API) to send database data stored on a raw logical volume to ADSM, or you could create a file, using either a database utility or the *dd* command, and then use ADSM commands to back up the file.

ADSMPIPE provides a direct interface between standard input file and ADSM. Any data that can be sent to the standard input of *adsmpipe* is sent on to ADSM through the ADSM API. No intermediate file is required.

ADSMPIPE aids in the backup of databases installed on raw devices. As shown in Figure 22 on page 54, ADSMPIPE reads raw devices and sends the data to ADSM using the ADSM API. It is an offline, full backup alternative. It is similar to using the dd command except that an intermediate file is not created. ADSMPIPE should be used *only* if no other better API alternative is available.

Oracle and ADSMPIPE

You can use ADSMPIPE for online backups of Oracle if you use Oracle's *alter tablespace begin backup* and *alter tablespace end backup* commands. Back up all of the raw logical volumes for each tablespace between the alter tablespace begin and end backups. Then back up the redo logs. See 6.7, "Partial Online Backup and Recovery with ADSM" on page 95 and 6.8, "Full Online Backup and Recovery with ADSM" on page 97 for details on how to use the alter tablespace commands.

The code was written to work with the ADSM Version 1.2 API but has also been tested with the ADSM Version 2.1 API. Unless you are running a Version 1 backup/archive client, we strongly recommend you use the Version 2.1 API.

See G.7, "International Technical Support Organization on the Internet" on page 579 for instructions on how to obtain the code by FTP from the Internet.

Before you can install ADSMPIPE you must have the ADSM API installed as well as a C compiler, as the code is provided as source code. The ADSM API return codes are documented in *ADSM: Using the API*, SH26-4002.

C.1 Installation

To install the adsmpipe code follow these steps:

1. Create a directory for the adsmpipe code. We suggest */usr/adsmpipe*. The code requires approximately 4M of space.

2. Using anonymous ftp download the tar file and rename it to adsmpipe.tar.Z

```
> ftp ftp.almaden.ibm.com
> cd SG244335
> binary
> get ADSMPIPE.TAR adsmpipe.tar.Z
> ascii
> get READ.ME
```

   It is important that the download file is rename with a .Z extension.

3. Uncompress the adsmpipe.tar.Z file:

```
> uncompress adsmpipe.tar.Z
```

   The uncompressed file is renamed as adsmpipe.tar.

4. Use the tar command to extract the contents of adsmpipe.tar:

```
>tar -xvf adsmpipe.tar
  x aix/ver1/aixapi.v1r2m7.README, 968 bytes, 2 media blocks.
  x aix/ver1/aixapi.v1r2m7.smit, 665600 bytes, 1300 media blocks.
  x aix/ver2/libApiDS.a, 427682 bytes, 836 media blocks.
  x aix/ver2/dscameng.txt, 268608 bytes, 525 media blocks.
  x aix/ver2/dsmapitca, 148698 bytes, 291 media blocks.
  x aix/adsmpipe/Makefile, 272 bytes, 1 media blocks.
  x aix/adsmpipe/README, 26626 bytes, 53 media blocks.
  x aix/adsmpipe/adsmblib.c, 31446 bytes, 62 media blocks.
  x aix/adsmpipe/adsmpipe.c, 7185 bytes, 15 media blocks.
  x aix/db2bkdel/db2bkdel, 34696 bytes, 68 media blocks.
  x solaris/ver1/sunapi.v1r2m7.README, 424 bytes, 1 media blocks.
  x solaris/ver1/sunapi.v1r2m7.tar.Z, 812453 bytes, 1587 media blocks.
  x solaris/ver2/dscameng.txt, 268608 bytes, 525 media blocks.
  x solaris/ver2/dsmapitca, 217032 bytes, 424 media blocks.
  x solaris/ver2/libApiDS.so.1.sol2, 393680 bytes, 769 media blocks.
  x solaris/adsmpipe/Makefile, 381 bytes, 1 media blocks.
  x solaris/adsmpipe/README, 27426 bytes, 54 media blocks.
  x solaris/adsmpipe/adsmblib.c, 31446 bytes, 62 media blocks.
  x solaris/adsmpipe/adsmpipe.c, 7185 bytes, 15 media blocks.
```

   When you issue the *tar -xvf adsmpipe.tar* command, the files are stored beneath the directory from which you issued the tar command. Make sure that you do not have any files with the same name because the tar command will overwrite them.

5. Compile the source code

   You now have to compile the adsmpipe program, using the *makefile* provided:

```
>make adsmpipe
  make
  cc -c adsmpipe.c
  cc -c adsmblib.c
  cc -o adsmpipe adsmpipe.o adsmblib.o libApiDS.a -L.
```

If you run the makefile from the adsmpipe directory and it does not work, you may have to copy the files in the adsmpipe directory to the /usr/lpp/adsm/api/bin directory.

API File Locations

The ADSM API file locations differ according to client platform and operating system level. See Appendix E, “ADSM API Files: Location and Environment Variables” on page 569 for API file locations.

C.2 Running adsmpipe

Program usage is similar to that of many other programs. Data to be stored goes to standard input. Data to be retrieved comes from standard output.

To make it easier to track the use of the adsmpipe program, you can associate a description of up to 256 characters with a file.

For files greater than 1 MB you must give ADSM information about the size of your file to ensure that it has space available for the data to be sent. The limit is given using **-l**. For example:

```
tar -cvf- $HOME | adsmpipe -c -f 'myhome.tar' -l 90M
```

will ensure that space for 90MB is available. If you do not give the size argument, the following message may be produced:

```
error in dsmSendData. rcMsg=(157)    The transaction will be aborted.
```

Figure 373 on page 556 shows the program used to back up, query, and restore a raw logical volume. In this case the volume is a file system created specifically for this purpose. It could, however, be any logical volume. You could, for example, back up your boot logical volume!

In this example we use the AIX *df* command to show the use of the logical volume and the AIX *lsvg* command to show that the volume is not in use when the adsmpipe program is run. The *mount* and *unmount* commands are used to start and stop users from accessing the file system.

Note: When you are restoring a JFS that you have backed up as a raw logical volume, you must first re-create it, using SMIT. This will create the basic JFS structure and the appropriate entries in */etc/filesystems*. You can then restore to the created logical volume and mount your file system.

For database backup you must make sure that your database has been shut down.

```
>mount /pipetest
>df /pipetest
Filesystem    Total KB    free %used   iused %iused Mounted on
/dev/lv17         4096    1264   69%      45     4% /pipetest

>lsvg -l rootvg | grep pipetest
lv17                jfs        1     1    1    open/syncd    /pipetest

>unmount /pipetest
>lsvg -l rootvg | grep pipetest
lv17                jfs        1     1    1    closed/syncd  /pipetest

>adsmpipe -cvf pipetest -l 8M < /dev/rlv17
adsm: about to send data to ADSM
adsm: sent 262144 bytes to ADSM
adsm: sent 524288 bytes to ADSM
:
adsm: sent 3932160 bytes to ADSM
adsm: sent 4194304 bytes to ADSM
adsm: committing transaction

>mount /pipetest
>rm /pipetest/*
>ls -al /pipetest
total 16
drwxr-sr-x   2 sys      sys         1024 Sep 25 13:43 .
drwxr-xr-x  24 bin      bin         1024 Sep 25 13:34 ..

>df /pipetest
Filesystem    Total KB    free %used   iused %iused Mounted on
/dev/lv17         4096    3936    3%       16     1% /pipetest

>unmount /pipetest
>adsmpipe -tvf pipetest
root     (A) 1995/09/25 13:42:04 /pipetest

>adsmpipe -xvf pipetest > /dev/rlv17
adsm: about to get data
adsm: read 262144 bytes from ADSM
adsm: read 524288 bytes from ADSM
:
adsm: read 3932160 bytes from ADSM
adsm: read 4194304 bytes from ADSM
adsm: read completed

>mount /pipetest
>df /pipetest
Filesystem    Total KB    free %used   iused %iused Mounted on
/dev/lv17         4096    1264   69%      45     4% /pipetest
```

Figure 373. ADSMPIPE Sample Session

Figure 374 on page 557 and Figure 375 on page 558 shows a sample backup of a real database logical volume. Figure 374 on page 557 shows the backup of the volume. Figure 375 on page 558 shows the deletion of the volume and its restore. In this case the database is Sybase.

Note: Some of the messages issued during this example have been edited out.

This Sybase instance has been installed with two logical volumes, */dev/rsybdb* and */dev/rsybsp.* The sample database *pubs2* has been installed.

```
1> select au_id, au_fname, au_lname from authors
2> go
 au_id       au_fname             au_lname
 ----------- -------------------- ----------------------------------------
 172-32-1176 Johnson              White
 213-46-8915 Marjorie             Green
 238-95-7766 Cheryl               Carson
      :
 899-46-2035 Anne                 Ringer
 998-72-3567 Albert               Ringer

(23 rows affected)
1> shutdown SYB_BACKUP
2> go
Backup Server: 3.48.1.1: The Backup Server will go down immediately.
Terminating sessions.
1> shutdown
2> go
Server SHUTDOWN by request.
The SQL Server is terminating this process.
DB-LIBRARY error:
        Unexpected EOF from SQL Server.
> exit
# ./adsmpipe -cvf /lvdump -l 20M < /dev/rsybdb
adsm: about to send data to ADSM
adsm: sent 262144 bytes to ADSM
adsm: sent 524288 bytes to ADSM
adsm: sent 786432 bytes to ADSM
:
adsm: sent 20709376 bytes to ADSM
adsm: sent 20971520 bytes to ADSM
adsm: committing transaction
# lsvg -l rootvg | grep sybdb
sybdb               jfs        5     5    1    closed/syncd  N/A
```

Figure 374. ADSMPIPE Database Logical Volume Backup

```
# rmlv -f sybdb
rmlv: Logical volume sybdb is removed.
# mklv -y'sybdb' rootvg 5
sybdb
# chown sybase /dev/sybdb
# chown sybase /dev/rsybdb
# ./adsmpipe -xvf /lvdump > /dev/rsybdb
adsm: about to get data
adsm: read 262144 bytes from ADSM
adsm: read 524288 bytes from ADSM
adsm: read 786432 bytes from ADSM
:
adsm: read 20709376 bytes from ADSM
adsm: read 20971520 bytes from ADSM
adsm: read completed
# su - sybase
$ cd install
$ startserver -f RUN_sybase
00:95/10/10 15:20:35.69 kernel  Using config area of disk for boot information
00:95/10/10 15:20:35.79 kernel  Using config area from primary master device.
  :
00:95/10/10 15:20:43.67 server  Recovery complete.
00:95/10/10 15:20:43.75 server  SQL Server's default sort order is:
00:95/10/10 15:20:43.83 server          'bin_iso_1' (ID = 50)
00:95/10/10 15:20:43.91 server  on top of default character set:
00:95/10/10 15:20:43.99 server          'iso_1' (ID = 1).
$ startserver -f RUN_SYB_BACKUP
Backup Server/10.0.2/P/RS6000SMP/AIX 3.2.5/1/OPT/Wed May 24 10:43:53 PDT 1995
Confidential property of Sybase, Inc.
(c) Copyright Sybase, Inc.  1987, 1995
All rights reserved.

Logging Backup Server messages in file '/u/sybase/install/backup.log'
$ isql -Usa -P -Ssybase
1> use pubs2
2> go
1> select au_id, au_fname, au_lname from authors
2> go
 au_id       au_fname             au_lname
 ----------- -------------------- ----------------------------------------
 172-32-1176 Johnson              White
 213-46-8915 Marjorie             Green
 238-95-7766 Cheryl               Carson
      :
 899-46-2035 Anne                 Ringer
 998-72-3567 Albert               Ringer

(23 rows affected)
```

Figure 375. ADSMPIPE Database Logical Volume Deletion and Restore

C.3 Changing the ADSM Password

You can use *adsmpipe* to change the ADSM client node password. You will only be able to change the password if you are running *adsmpipe* as root.

C.3.1 Passwordaccess=Prompt

If you have the passwordaccess parameter set to prompt, use the -p option as follows:

- `adsmpipe -p password`

  This option logs on to the server only.

- `adsmpipe -p oldpw/newpw/newpw`

  This option changes the password from oldpw to newpw.

- `adsmpipe -cf /adsmpipe -p password < filename`

  This option logs on to the server and backs up the file.

C.3.2 Passwordaccess=Generate

If you have the passwordaccess parameter set to generate, use the -p option as follows:

- `adsmpipe -p password`

  This option generates the password file on the client.

- `adsmpipe -p oldpw/newpw/newpw`

  This option changes the password from oldpw to newpw and generates the password file if it does not already exist. You should not have to generate the file this way. Use adsmpipe -p password.

- `adsmpipe -cf /adsmpipe -p password < filename`

  This option generates the password file and backs up the file.

Note: When you have passwordaccess set to generate, you only have to set the password initially because ADSM will change it automatically when it expires.

Figure 376 on page 560 shows the full set of options for *adsmpipe*.

```
adsmpipe -[tcxd] -f<filename> [-l<size>][-v][-s<filespace>]

Create, extract or lists files in the ADSM pipe backup area
   -t      Lists files in pipe backup area matching the pattern
   -c      Creates a file in pipe backup area.
            Data comes from standard input.
   -x      Extracts a file in pipe backup.
            Data goes to standard output.
   -d      Deletes the file from active store.

   -B      Store data in the backup space of the ADSM server (Default)
   -A      Store data in the archive space of the ADSM server

   -f <file>
           Provides filename for create, and extract operations.
            For list operations, the filename can be a pattern
   -l <size>
           Estimated size of data in bytes when creating.
   -p <oldpw/newpw/newpw>
           Changes the password from oldpw to newpw.
      <passwd>
           Initialises password file for ADSM API with passwd.
   -v      Verbose
   -n <count>
           File number to retrieve if multiple versions
   -s <filespace>
           Specify file space (default "/adsmpipe")
```

Figure 376. ADSMPIPE Command Arguments

The -l option is required because ADSM must know in advance the amount of data that will be sent to the ADSM server. ADSM uses this information to decide which storage pools to use and to perform any necessary migration to create enough space for the data when it is sent. The -l option does not take ADSM compression into account.

In an ideal world, adsmpipe would be able to tell how much data you actually have. Unfortunately, adsmpipe cannot determine the actual size of the data because the data is coming from standard input, so the size is not known in advance. Buffering the data is not a reasonable option because the file might be enormous.

You can use several approaches with the -l option:

- Do not specify it at all.

  The default will then be 1MB. If you have an empty storage pool to which to send the data, or if the data is less than 1MB, you do not have to specify the -l option.

- Specify a large value.

  This will cause the contents of the storage pool to be flushed (due to migration) to make room for the data.

- Try a value and then double it until the data is sent successfully.

  This approach requires an input stream that is rewindable, such as disk space.

In all three approaches, the -l option refers to the space occupied after compression if you have turned compression on. Therefore, the value you select should be set to the largest possible value for the data if you want to guarantee completion of the adsmpipe command.

Appendix D. ADSM Database Matrices

Figure 377 through Figure 382 on page 568 show the various backup methods that can be applied to the database products reviewed in this book.

DB2/2 backup alternatives with ADSM

Backup alternatives	Database mode	Intermediate OS/2 file created	Backup granularity	Platforms Supported
DB2/2 V1.1 backup	offline	yes	database, changes only	OS/2
DB2/2 V1.1 backup with user exit program	offline	yes	database	OS/2
ADSM directly	offline	no	database	OS/2, NT
DB2/2 V1.2 backup	offline, quiesced	yes	database, changes only	OS/2
DB2/2 V1.2 backup with user exit program	offline, quiesced	yes	database	OS/2
DB2/2 V2 backup using ADSM API	online, offline	no	database, tablespace	OS/2, NT

DB2MAT

Figure 377. DB2/2 Backup Alternatives with ADSM

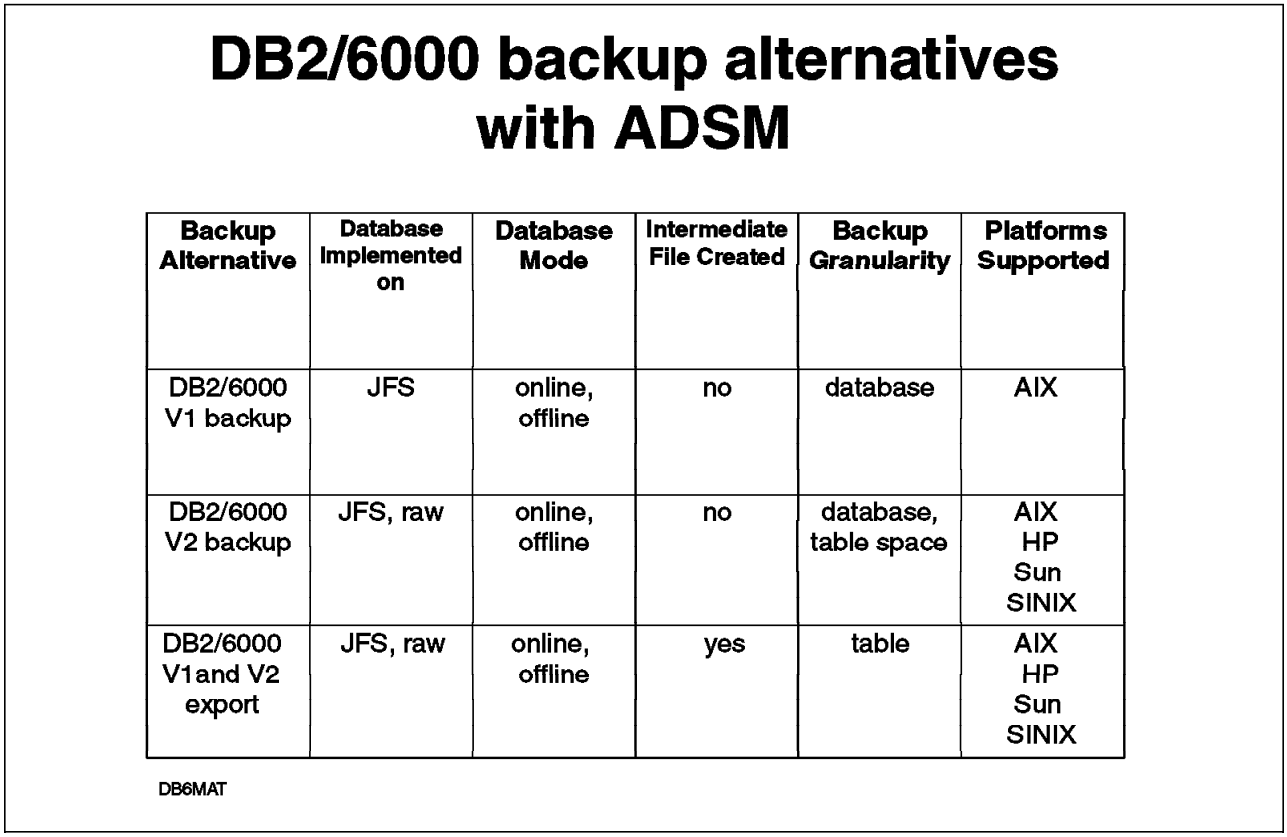


Figure 378. DB2/6000 Backup Alternatives with ADSM

INFORMIX-OnLine backup alternatives with ADSM

Backup Alternative	Database Implementation	Database Mode	Intermediate File Created	Backup Granularity	Platforms Supported
dd	raw device	offline	yes	raw device	any
ADSMPIPE	raw device	offline	no	raw device	any
INFORMIX dbexport	raw device, file system	online	yes	database	any
Tbunload/ Onunload	raw device, file system	online	yes	database, table	any
Tbtape/ Ontape	raw device, file system	online, quiesced	yes	database, incremental	any
Onarchive	raw device, file system	online, quiesced	yes	database, dbspace, incremental	any
ADSM directly	file system	offline	no	database	any
ON-Bar using ADSM X/OPEN API	raw device file system	online, quiesced	no	database, dbspace, incremental	AIX Solaris
Future: DataTools SQL-BackTrack	raw device, file system	online, quiesced	no	database, dbspace, incremental	AIX, HP, Sun

INFMAT

Figure 379. INFORMIX-OnLine Backup Alternatives with ADSM

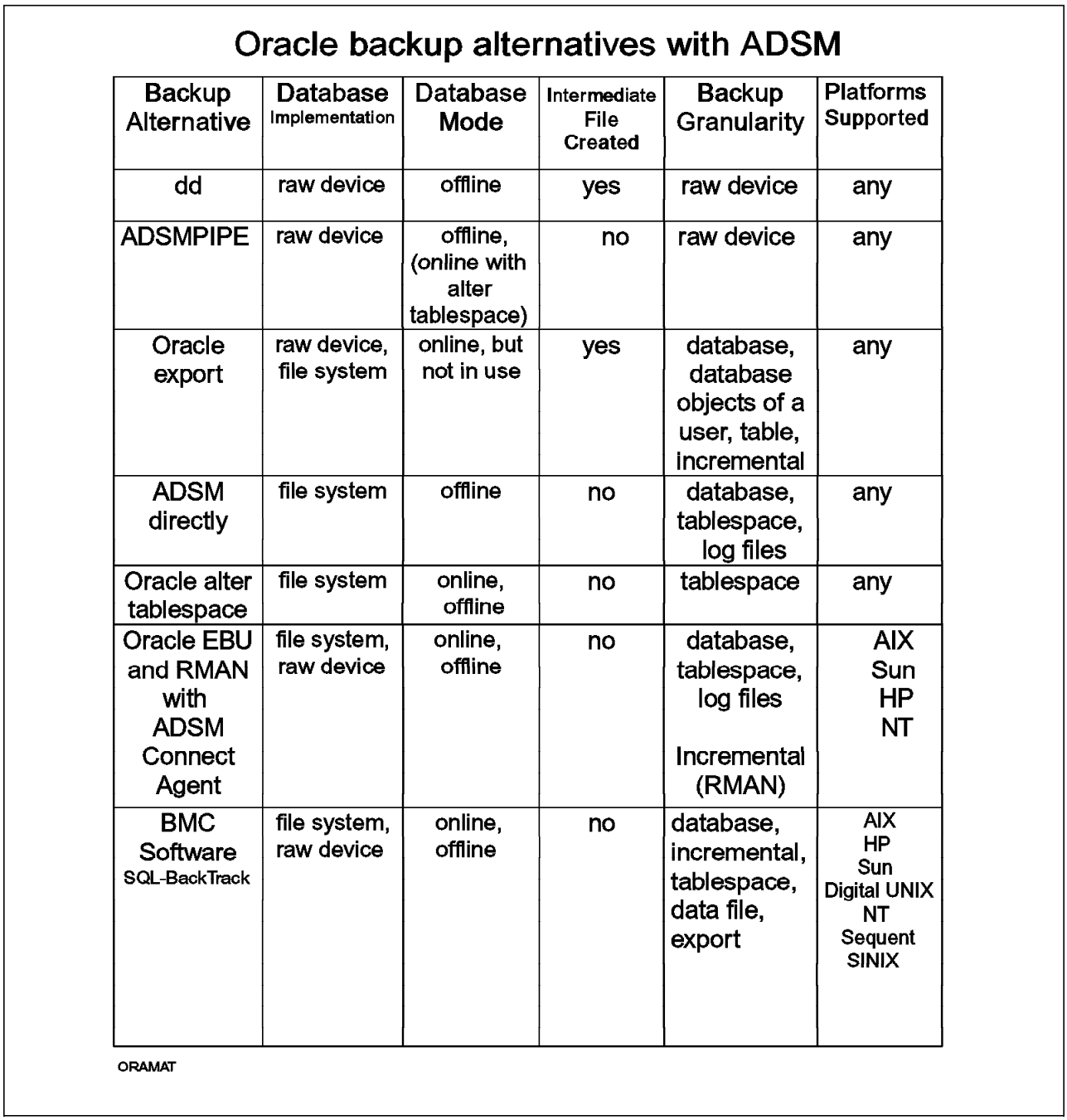


Figure 380. Oracle Backup Alternatives with ADSM

Sybase backup alternatives with ADSM

Backup Alternative	Database Implemented on	Database Mode	Intermediate File Created	Backup Granularity	Platforms Supported
dd	raw device	offline	yes	raw device	any
ADSMPIPE	raw device	offline	no	raw device	any
dump database	raw device, file system	online, offline	yes	database, transaction logs	any
dump transaction	raw device, file system	online	yes	transaction logs	any
bulk copy utility	raw device, file system	online	yes	table	any
ADSM directly	file system	offline	no	database	any
SQL-BackTrack using ADSM API	raw device, file system	online, offline	no	database, database objects, incremental	AIX HP Sun Soon: Digital UNIX NCR UNIX

SYBMAT

Figure 381. Sybase Backup Alternatives with ADSM

SAP R/3 backup alternatives with ADSM

Database	Database Implementation	Database Mode	Intermediate File Created	Supported Backup Alternative	Platforms Supported
Oracle OPS	file system, raw device raw device	offline, online offline, online	no no	BRBACKUP and BRARCHIVE using BACKINT/ADSM Soon: BACKINT/ADSM	AIX Digital UNIX HP NCR UNIX SINIX Solaris Soon: NT
INFORMIX-OnLine	raw device	offline, online	yes	OnArchive Soon: ON-Bar	AIX HP Solaris
DB2/6000 DB2PE	file system	offline, online	no	DB2/6000 Backup using ADSM API DB2admin tool	AIX HP NT SINIX Solaris
ADABAS-D	raw device	online, offline	no	ADINT/ADSM	AIX HP NT SINIX Solaris On-request: Digital UNIX NCR UNIX

SAPMAT

Figure 382. SAP R/3 Backup Alternatives with ADSM

Appendix E. ADSM API Files: Location and Environment Variables

The location of ADSM API files and the default value for the DSMI_DIR environment variable differ according to the client platform and operating system level you are using. Here are the locations and values for AIX, HP-UX, SUN, Solaris, SINIX, Digital UNIX, Sequent, and NT.

E.1 AIX 3.x and 4.x

The locations of the API files are:

```
/usr/lpp/adsm/bin
                    dsm.sys
                    dsmapitca
                    en_US/dsmclient.cat
/usr/lpp/adsm/api/bin
                    libApiDS.a (installed with a link to /usr/lib)
                    libXApi.a (X/Open library)
```

The default value for DSMI_DIR is /usr/lpp/adsm/bin.

INFORMIX-OnLine ON-Bar users must define a symbolic link between the X/Open library and /usr/lib/ibsad001.a.

E.2 HP-UX V9 and V10

The locations of the API files are:

```
/usr/adsm
                    dsm.sys
                    dsmapitca
                    dscameng.txt
/usr/adsm/api
                    libApiDS.sl
```

Define a symbolic link between the API library and /usr/lib/libApiDS.sl.

The default value of the DSMI_DIR environment variable is /usr/adsm.

E.3 SUN V4

The locations of the API files are:

```
/usr/adsm
                    dsm.sys (real file)
                    dscameng.txt (real file)
/usr/adsm/sun4
                    dsmapitca
                    dscameng.txt (link)
                    dsm.sys (link which customer defines)
/usr/adsm/api
                    libApiDS.so.1.0.sun41
```

Define a symbolic link between /usr/adsm/sun4/dsm.sys and the real dsm.sys file and between the API library and /usr/lib/libApiDS.so.1.0.

There is no default value for the DSMI_DIR environment variable. Set DSMI_DIR to /usr/adsm/sun4.

E.4 Solaris V2.3 and V2.4

The locations of the API files are:

```
/usr/adsm or /opt/IBMDSMapi
                    dsm.sys (real file)
                    dscameng.txt (real file)
/usr/adsm/solaris or /opt/IBMDSMapi/solaris
                    dsmapitca
                    dscameng.txt (link)
                    dsm.sys (link that you must define)
/usr/adsm/api or /opt/IBMDSMapi/api
                    libApiDS.so.sol2
```

If you install this client, using a tar file, the install path is /usr/adsm. If you install this client, using a pkgadd file, the install path is /opt/IBMDSMapi.

Define a symbolic link between $DSMI_DIR/dsm.sys and the real dsm.sys file and between the API library and /usr/lib/libApiDS.so.

There is no default value for the DSMI_DIR environment variable. Set DSMI_DIR to /usr/adsm/solaris or /opt/IBMDSMapi/solaris.

E.5 Solaris V2.5

The locations of the API files are:

```
/usr/sbin
                    dsm.sys (real file)
/opt/IBMDSMap5/en_US
                    dsmclient.cat (real file)
/opt/IBMDSMap5/solaris
                    dsmapitca
                    en_US/dsmclient.cat (link)
                    dsm.sys (link)
/opt/IBMDSMap5/api
                    libApiDS.so.sol2
```

Define a symbolic link between the API library and /usr/lib/libApiDS.so.

The default value for the DSMI_DIR environment variable is /usr/sbin. Set it to /opt/IBMDSMap5/solaris.

E.6 SINIX RISC

The locations of the API files are:

```
/usr/adsm
                    dsm.sys
                    dscameng.txt
                    dsmapitca
/usr/adsm/api
                    libApiDS.so
```

The default value for the DSMI_DIR environment variable is /usr/adsm.

E.7 Digital UNIX

The locations of the API files are:

```
/usr/opt/adsm
                    dsm.sys
                    dscameng.txt
                    dsmapitca
/usr/opt/adsm/api
                    libApiDS.so
```

The default value for the DSMI_DIR environment variable is /usr/opt/adsm.

E.8 Sequent

The locations of the API files are:

```
/usr/adsm
                    dsm.sys
                    dscameng.txt
                    dsmapitca
/usr/adsm/api
                    libApiDS.a
```

The default value for the DSMI_DIR environment variable is /usr/adsm.

E.9 NT

The locations of the API files are:

```
<path>\baclient
                    dsm.opt
<path>\api\bin
                    adsm32.dll
                    dscameng.txt
<path>\api\dll
                    blkhook.dll

                  If the customer's application uses blkhook.dll, the
                  DLL must be copied to <path>\api\bin to consolidate
                  the files for the PATH reference.

  The default path (if you have not previously installed the NT API) is:
     c:\win32app\ibm\adsm
```

Add <path>\api\bin to PATH for the DLLs. There is no default value for the DSMI_DIR environment variable. Set it to <path>-api-bin. Set DSMI_CONFIG to <path>\baclient\dsm.opt.

Appendix F. Special Notices

This publication is intended to help IBM, customer, vendor, and consultant personnel back up workstation databases using ADSTAR Distributed Storage Manager (ADSM). The information in this publication is not intended as the specification of any programming interfaces that are provided by ADSM. See the PUBLICATIONS section of the IBM Programming Announcement for ADSM for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

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, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (″vendor″) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other

operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

ADSTAR
AIX
AIX/6000
AIXwindows
AS/400
BookManager
BookMaster
DATABASE 2
DataHub
DB2
Enterprise Systems Architecture/390
IBM
InfoExplorer
MVS/ESA
Operating System/2
Operating System/400
OS/2
OS/390
OS/400
PowerPC Architecture
POWERserver
POWERstation
PS/2
RISC System/6000
RS/6000
S/390
Scalable POWERparallel Systems
S/390 Parallel Enterprise Server
VM/ESA

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

Java and HotJava are trademarks of Sun Microsystems, Incorporated.

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

Pentium, MMX, ProShare, LANDesk, and ActionMedia are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names may be trademarks or service marks of others.

Appendix G. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

G.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How to Get ITSO Redbooks” on page 581.

Book Title	Publication Number
General Topics	
ADSM Concepts	SG24-4877
Using ADSM Hierarchical Storage Management	SG24-4631
Client Disaster Recovery: Bare Metal Restore	SG24-4880
ADSM Version 3 Technical Guide	SG24-2236
Specific Server Books	
ADSM Server for Windows NT Configuration and Recovery Examples	SG24-4878
ADSM Server-to-Server Implementation and Operation	SG24-5244
Getting Started with ADSM/6000	GG24-4421
ADSM for AIX: Advanced Topics	SG24-4601
AIX Tape Management	SG24-4705
ADSTAR Distributed Storage Manager/6000 on 9076 SP2	GG24-4499
ADSM for MVS: Recovery and Disaster Recovery	SG24-4537
ADSM for MVS: Using Tapes and Tape Libraries	SG24-4538
Getting Started with ADSM/2	GG24-4321
ADSM for OS/2: Advanced Topics	SG24-4740
Setting Up and Implementing ADSTAR Distributed Storage Manager/400	GG24-4460
ADSM/VSE Implementation Guide	SG24-4266
Specific Client Books	
Getting Started with ADSM NetWare Clients	GG24-4242
Getting Started with ADSM AIX Clients	GG24-4243
Windows NT Backup and Recovery with ADSM	SG24-2231
ADSM API Examples for OS/2 and Windows	SG24-2588
ADSM with Other Products	
Using ADSM to Back Up Lotus Notes	SG24-4534
Hierarchical Storage Management for NetWare: ADSM and AvailHSM Implementation	SG24-4713
Using ADSM to Back Up OS/2 LAN Server and Warp Server	SG24-4682
Backup, Recovery, and Availability with DB2 Parallel Edition on RISC/6000	SG24-4695
ADSM Operation and Management with TME10	SG24-2214

G.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
Lotus Redbooks Collection	SBOF-6899	SK2T-8039
Tivoli Redbooks Collection	SBOF-6898	SK2T-8044
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849

CD-ROM Title	Subscription Number	Collection Kit Number
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
RS/6000 Redbooks Collection (PDF Format)	SBOF-8700	SK2T-8043
Application Development Redbooks Collection	SBOF-7290	SK2T-8037

G.3 ADSM Product Publications

Book Title	Publication Number
ADSM V3R1 Messages	GC35-0271
ADSM V3R1 AIX Quick Start	GC35-0273
ADSM V3R1 AIX Administrator's Guide	GC35-0274
ADSM V3R1 AIX Administrator's Reference	GC35-0275
ADSM V3R1 MVS Quick Start	GC35-0276
ADSM V3R1 MVS Administrator's Guide	GC35-0277
ADSM V3R1 MVS Administrator's Reference	GC35-0278
ADSM V3R1 AIX License	SC35-0283
ADSM V3R1 MVS License	GC35-0284
ADSM V3R1 Windows NT Administrator's Guide	GC35-0292
ADSM V3R1 Windows NT Administrator's Reference	GC35-0293
ADSM V3R1 Windows NT License	SC35-0294
ADSM V3R1 Windows NT Quick Start	GC35-0295
ADSM V3R1 Using the UNIX Backup-Archive Client	SH26-4075
ADSM V3R1 Using the OS/2 Backup-Archive Client	SH26-4076
ADSM V3R1 Using the Novell NetWare Backup-Archive Client	SH26-4077
ADSM V3R1 Using the Microsoft Windows Backup-Archive Client	SH26-4078
ADSM V3R1 Using the Apple Macintosh Backup-Archive Client	SH26-4079
ADSM V3R1 Installing the Clients	SH26-4080
ADSM V3R1 Using the Application Programming Interface	SH26-4081
ADSM V3R1 Trace Facility Guide	SH26-4082
ADSM V3R1 Client Reference Cards	SX26-6019
ADSMConnect Agent for Notes Backup on OS/2	SH26-4047
ADSMConnect Agent for Oracle7 Backup on AIX Installation and User's Guide	SH26-4061
ADSMConnect Agent for Oracle Backup on Sun Solaris Installation and User's Guide	SH26-4063
ADSMConnect Agent for Lotus Notes on Windows NT Installation and User's Guide	SH26-4065
ADSMConnect Agent for Lotus Notes on AIX Installation and User's Guide	SH26-4067
ADSMConnect Agent for Microsoft SQL Server Installation and User's Guide	SH26-4069

G.4 ADSM Online Product Library

All of the ADSM publications are available in online readable format on the CD-ROM listed below:

CD-ROM Title	Publication Number
ADSM V3R1.0 MVS Online Product Library	SK3T-1396

G.5 Other Publications

These publications are also relevant as further information sources:

- Oracle Publications
  - *Oracle7 Version 7.3.4 On-Line Generic Documentation CD-ROM*, Part No. A57673-01
  - *Oracle8 Version 8.0.3 On-Line Generic Documentation CD-ROM*, Part No. A55125-01
- INFORMIX-OnLine Publications
  - *INFORMIX-OnLine Version 5.0 Administrator's Guide*, Part No. 000-7106
  - *INFORMIX-OnLine Version 6.0 Administrator's Guide, Volume 1*, Part No. 000-7667
  - *INFORMIX-OnLine Version 6.0 Administrator's Guide, Volume 2*, Part No. 000-7668
  - *INFORMIX-OnLine Version 6.0 Archive and Backup Guide*, Part No. 000-7591
  - *Guide to 7.1 Feature Enhancements*, Part No. 000-7748
  - *INFORMIX-OnLine Version 7.1 Administrator's Guide, Volume 1*, Part No. 000-7778
  - *INFORMIX-OnLine Version 7.1 Administrator's Guide, Volume 2*, Part No. 000-7777
  - *INFORMIX-OnLine Version 7.1 Archive and Backup Guide*, Part No. 000-7627
  - *INFORMIX-OnLine Version 7.1 On-Archive Quick Start Guide*, Part No. 000-7775
  - *INFORMIX-OnLine Version 7.21 Dynamic Server Backup and Restore Guide*, Beta verison
  - *INFORMIX-OnLine XPS Backup and Restore Guide*, Part No. 000-743
  - *INFORMIX-OnLine XPS Version 8 Product Briefing, July 1995*
- Sybase Publications
  - *Sybase Installation Guide for IBM RISC System/6000 AIX* , Document ID 34260-01-1000-02
  - *What's New in Sybase SQL Server Release 10*, Document ID 36440-01-1000-03
  - *Sybase SQL Server Utility Programs for UNIX*, Document ID 30475-01-1000-03
  - *System Administration Guide for Sybase SQL Server*, Document ID 32500-01-1000-02
  - *System Administration Guide Supplement for IBM RISC System/6000 AIX*, Document ID 35290-01-1000-02
  - *Sybase SQL Server Reference Volume I: Commands, Functions, and Topics*, Document ID 32401-01-1000-02
  - *Sybase SQL Server Reference Volume II: System Procedures and Catalog Stored Procedures*, Document ID 32402-01-1000-02

- DB2/6000 Publications
  - *Version 1 Information and Planning Guide*, GC09-1569
  - *Version 1 Administration Guide*, SC09-1571
  - *Version 1 Command Reference*, SC09-1575
  - *Version 2 Information and Concepts Guide*, S20H-4664
  - *Version 2 Installation and Operation Guide*, S20H-4757
  - *Version 2 Planning Guide*, S20H-4758
  - *Version 2 Administration Guide*, S20H-4780
  - *Version 2 Installing and Using UNIX AIX Clients*, S20H-4666
  - *Version 2 Command Reference*, S20H-4645
- DB2/2 Publications
  - *Version 1 Information and Planning Guide*, S62G-3662
  - *Version 1 Install Guide*, S62G-3664
  - *Version 1 DB2/2 Guide*, S62G-3663
  - *Version 1 Command Reference*, S62G-3670
  - *Version 2 Information and Concepts Guide*, S20H-4664
  - *Version 2 Installation and Operation Guide*, S20H-4785
  - *Version 2 Planning Guide*, S20H-4784
  - *Version 2 Administration Guide*, S20H-4780
  - *Version 2 Installing and Using OS/2 Clients*, S20H-4782
  - *Version 2 Command Reference*, S20H-4645
- DataHub Publications
  - *DataHub for OS/2 Version 2 Installation and Configuration* SC26-8366
  - *Using DataHub for OS/2 Version 2*, SC26-8350
  - *DataHub/2 Command Reference Release 2*, SC26-3044-01
  - *DataHub for UNIX Operating Systems Release 1 Installation and Configuration*, 33H1617
  - *DataHub for UNIX Operating Systems Release 1 User's Guide*, SC26-8669
- SQL-BackTrack Publications
  - *SQL-BackTrack for Oracle Installation Notes, Version 3.0.0 and 3.0.1*
  - *SQL-BackTrack for Oracle Release Notes, Version 2.1.3 Performance Release*
  - *SQL-BackTrack for Sybase Release Notes, Version 2.1.2*
  - *SQL-BackTrack for Sybase Release Notes, Version 2.1.3*
  - *SQL-BackTrack for Sybase Release Notes, Version 3.0.1*
  - *SQL-BackTrack for Sybase Version 2.1.2 Installation Notes*
  - *SQL-BackTrack for Sybase Version 2.1.3 Installation Notes*
  - *SQL-BackTrack for Sybase Version 3.0.1 Installation Notes*
  - *SQL-BackTrack for Oracle User's Guide, Version 2.0*

- *SQL-BackTrack for Sybase User's Guide, Version 2.1*
- *SQL-BackTrack for Sybase User's Guide, Version 3.0*
- *SQL-BackTrack ADSM OBSI Module User's Guide, Version 2.1*
- *SQL-BackTrack ADSM OBSI Module Version 2.1 Installation Notes*

• SAP R/3 and BACKINT Publications

- *SAP Database Administration: Oracle, Using BRBACKUP and BRARCHIVE*
- *BACKINT/ADSTAR Distributed Storage Manager Program Description and Operations*, SC33-6379

G.6 ITSO Redbooks on the World Wide Web (WWW)

Internet users may find information about redbooks on the ITSO World Wide Web home page. To access the ITSO Web pages, point your Web browser (such as WebExplorer from the OS/2 3.0 Warp BonusPak) to the following:

```
http://www.redbooks.ibm.com/redbooks
```

IBM employees can access LIST3820s of redbooks as well. Point your web browser to the IBM Redbooks home page:

```
http://w3.itsc.pok.ibm.com/redbooks/redbooks.html
```

G.7 International Technical Support Organization on the Internet

If you do not have World Wide Web access, you can obtain the list of all current redbooks through the Internet by anonymous FTP to:

```
> ftp ftp.almaden.ibm.com
> cd /redbooks
> get itsopub.txt
```

The FTP server, *ftp.almaden.ibm.com*, also stores the ADSMPIPE sample code described in this redbook. To retrieve the sample files, issue the following commands from the */redbooks* directory:

```
> cd SG244335
> binary
> get ADSMPIPE.TAR adsmpipe.tar.Z
> ascii
> get READ.ME
```

All users of ITSO publications are encouraged to provide feedback to improve quality over time. In addition to using the feedback form in the back of this redbook, you can also send questions about and feedback on redbooks to:

REDBOOK at WTSCPOK *or*
REDBOOK@VNET.IBM.COM *or*
USIB5FWN at IBMMAIL

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at http://www.redbooks.ibm.com/.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Redbooks Web Site on the World Wide Web**

  http://w3.itso.ibm.com/

- **PUBORDER** — to order hardcopies in the United States
- **Tools Disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLCAT REDPRINT
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get BookManager BOOKs of redbooks, type the following command:

  ```
  TOOLCAT REDBOOKS
  ```

  To get lists of redbooks, type the following command:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  ```

  To register for information on workshops, residencies, and redbooks, type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1998
  ```

- **REDBOOKS Category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (http://www.redbooks.ibm.com/redpieces.html). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** — send orders to:

	IBMMAIL	**Internet**
In United States:	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

IBM Publications Publications Customer Support P.O. Box 29570 Raleigh, NC 27626-0570 USA	IBM Publications 144-4th Avenue, S.W. Calgary, Alberta T2P 3N5 Canada	IBM Direct Services Sortemosevej 21 DK-3450 Allerød Denmark

- **Fax** — send orders to:

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States)** or **(+1)001-408-256-5422 (Outside USA)** — ask for:

Index # 4421 Abstracts of new redbooks
Index # 4422 IBM redbooks
Index # 4420 Redbooks for last six months

- **On the World Wide Web**

Redbooks Web Site	http://www.redbooks.ibm.com/
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (`http://www.redbooks.ibm.com/redpieces.html`). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

- Invoice to customer number
- Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

ITSO Redbook Evaluation

Using ADSM to Back Up Databases
SG24-4335-03

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.ibm.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Which of the following best describes you?
__Customer __Business Partner __Solution Developer __IBM employee
__None of the above

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes____ No____

If no, please explain:

_____

_____

_____

_____

What other redbooks would you like to see published?

_____

_____

_____

Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

_____

_____

_____

_____

_____

SG24-4335-03
Printed in the U.S.A.