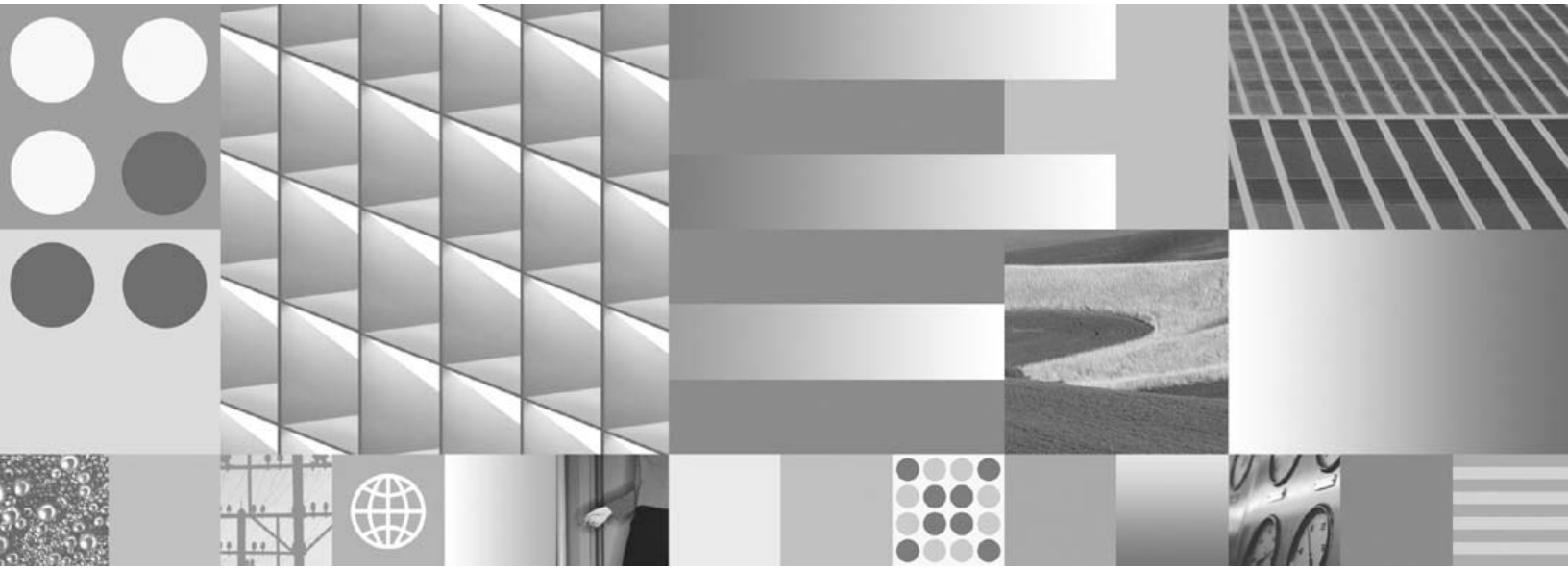
IBM FileNet Image Services

Version 4.1

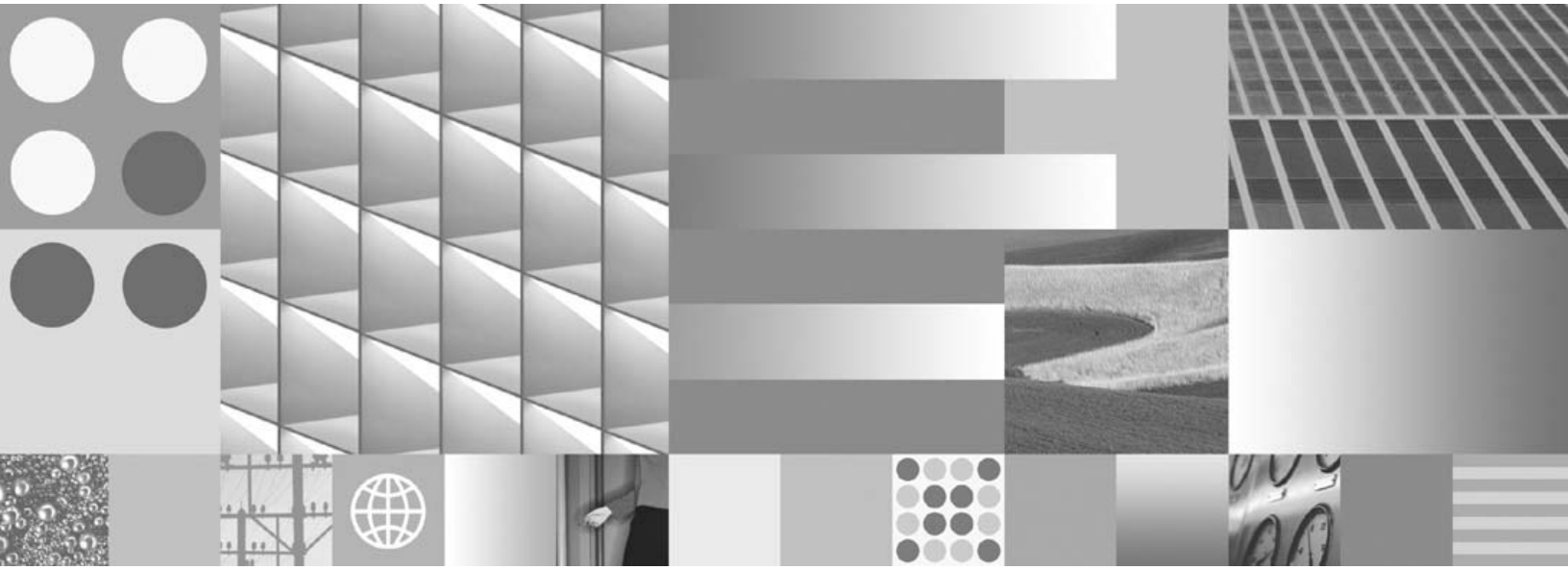


Enterprise Backup and Restore

GC31-5548-01

IBM FileNet Image Services

Version 4.1



Enterprise Backup and Restore

GC31-5548-01

Note
Before using this information and the product it supports, read the information in "Notices" on page 441.

This edition applies to version 4.1 of IBM FileNet Image Services (product number 5724-R95) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

About This Manual

The *FileNet Image Services Enterprise Backup/Restore User's Guide* is written for the FileNet Image Services (IS) System Administrator or other personnel responsible for backing up and restoring data on Image Services servers. This document describes Enterprise Backup/Restore (EBR), EBR concepts and functions, and how to use EBR to back up and restore a FileNet IS system.

Note In IS 4.1, EBR can be used to backup both Oracle 9i and Oracle 10g relational databases.

Chapters 1 and 2 introduce you to EBR and describe concepts and terms with which you should be familiar before you use EBR.

Information in **Chapter 3, "Quick Start," on page 73** helps you get started with EBR. Using sample scripts (coded programs that tell EBR how to back up or restore data), you can run simple EBR jobs before you implement your own customized scripts.

Other topics covered in this manual are:

- How to develop a backup strategy for your FileNet Image Services enterprise
- How to develop your own customized backup and restore scripts
- How to run EBR backup and restore programs

Appendixes contain more detailed information about the EBR programs, utilities, and issues you may have to address at your site before using EBR to back up and restore your system.

Accessing IBM FileNet Documentation

To access documentation for IBM® FileNet® products:

1. Navigate to the Information Management support page (**www.ibm.com/software/data/support**).
2. Select the appropriate IBM FileNet product from the “Select a category” list.
3. From the Product Support page, click Product Documentation under Learn.
4. From the Product Documentation page
   a. If necessary, click the Doc Link for the appropriate component product to display the document list.
   b. Click the icon in the appropriate release column to access the document you need.

Related Documents

The following documents provide additional information about your FileNet system.

- *System Administrator’s Handbook*
  - *System Administrator’s Companion for UNIX* or ...
  - *System Administrator’s Companion for Windows Server*
- *Oracle 9i Backup and Recovery Documentation Online Roadmap*
- *System Tools Reference Manual*

To download IBM FileNet documentation from the IBM support page, see **“Accessing IBM FileNet Documentation” on page 20**.

Document revision history

IS version	Date	Comment
4.1	June 2008	Documentation refresh.
4.1	Nov. 2007	Blue wash.
4.1	June 2007	Initial release.

Conventions Used in This Manual

We call your attention to information throughout this manual using the following conventions.

Cautions, Notes, and Tips

The following message types call attention to important information:

CAUTION Caution boxes signal possible unexpected consequences of an action, such as loss of data or time.

Note Note boxes draw your attention to essential information that you should be sure to read.

Tip Tip boxes introduce ideas that might make your work easier.

Typing Instructions

Instructions you enter at the command line are indented and appear in a bold typeface as shown in the example below:

```
EBR_label tape=/dev/rmt0 type=8mm \
    ser=BDT950909001 vg=BANDIT_MONPM3_ONLIVAL
```

For lengthy commands, each line following the first is indented.

Note When entering a lengthy command on a UNIX command line, you must also type a **backslash** (called a continuation character) at the end of all but the last line to indicate that the command continues.

Emphasis

Bold typeface within text emphasizes an individual word or phrase. Take special note of bold text, as in the following example:

> If you perform a full backup of a disk cache, you must perform a **full** backup of the transient database **in the same script**.

Syntax

Syntax definitions are indented from the body text:

```
EBR_script (format_level=1;);
```

Required Parameters

Parameters that require you to provide information are shown within angle brackets (< >):

EBR_label tape=<file> type=<device_type> –erasedata

When you must make a choice from among available options within the parameter, the available choices are separated by a bar within the angle brackets. In the following example, you must choose either true or false:

restore_control_file = <true | false>;

Options

Optional parameters and keywords are within square brackets:

EBR [–syntax] @<script> [<param> ...]

Left-side Flags

MultSv

Left-side flags indicate that the text applies to a particular type of server, such as multiple servers or combined server, or a type of platform, such as UNIX or Windows Server. An example is the MultSv flag to the left of this paragraph. The MultSv flag indicates information for users with more than one server. **WorkGroup users** and other users with single-server configurations need not read the sections with this flag.

Computer Output

Information that the system displays (such as script file contents, system messages, or output from program execution) is shown in the following manner:

```
EBR_script(format_level = 2; );
BACKUP_GLOBAL_OPTIONS
      ...
END_BACKUP_GLOBAL_OPTIONS
DATASETS
      ...
END_DATASETS
BACKUP_OPTIONS
      ...
END_BACKUP_OPTIONS
...
```

Lengthy output displays may not be enclosed in a box as shown above. For readability, the output may be delineated with a line above the start of displayed data and another at the end.

What to Read First

We suggest that you first read **Chapter 1, “Introduction,” on page 26** and **Chapter 2, “Understanding EBR Concepts,” on page 42**. Then, to get started with EBR right away, read **Chapter 3, “Quick Start,” on page 73**. The “Quick Start” chapter leads you through a backup and restore operation using sample EBR scripts.

After you become familiar with EBR through Quick Start, read the remaining chapters and appendixes for more in-depth information.

Software Education

IBM provides various forms of education. Please visit Global Learning Services on the IBM Web site at (**www-306.ibm.com/software/sw-training/**).

Comments and Suggestions

To send comments and suggestions regarding the IBM FileNet documentation, please e-mail your comments to **comments@us.ibm.com**. Be sure to include the name of the product, the version number of the product, and the name of the book (if applicable). If you are commenting on a specific text, include the location of the text (for example, a chapter and section title, a table number, a page number or a help topic title). Your suggestions help us improve the products we deliver.

1 Introduction

Image Services Enterprise Backup/Restore (EBR) is a set of **scripts** and programs designed to automate the process of protecting your data through backups and restores. Restores are performed infrequently, usually to recover from a disk failure or other disaster.

Before you attempt a backup or restore using EBR, you should read, at a minimum, “Introduction” (this chapter), **Chapter 2, “Understanding EBR Concepts,” on page 42**, and **Chapter 3, “Quick Start,” on page 73**. Other chapters contain detailed information about setting up a backup strategy for your environment and developing your scripts. Appendixes contain additional information about specific components or functions of the EBR product.

Who Can Run EBR?

Any user who is a member of **all three** of the following FileNet groups has permission to run EBR and EBR utilities:

fnusr
fnadmin
dba

Membership in these groups is established by the System Administrator during Image Services (IS) software installation.

A user must be a member of the dba group to restore Oracle® control files and archive redo logs using EBR. A user who has dba group membership is typically also a member of the Administrators group (on Windows® Server platform only) and the fnadmin group. However, this group assignment decision is made by the System Administrator.

To back up non-FileNet RDBMS databases, you must be logged on as a user that has backup permission. Refer to your RDBMS documentation for appropriate permission levels.

The following table describes the permission level required to back up or restore each of the supported dataset types.

Dataset Type	Required Permission
MKF	Must be member of fnadmin group.
Cache	Must be member of fnadmin group.
Partition	Must be root user or have appropriate permission setting for the partition. See your System Administrator for this information.
Oracle *	Must be member of dba group.

* EBR can back up Oracle 9i datasets only if they were created with a block size of 2 KB.

General Recommendations

Disk Mirroring and RAID

EBR is one of several methods you may use to protect your data. Other methods include **disk mirroring** and Redundant Array of Independent Disks (**RAID**). EBR provides an automated software solution to performing regular backups and, when necessary, restores. Disk

mirroring and RAID each employ combinations of hardware and software to provide data duplication (redundancy).

We **highly recommend** that you use disk mirroring or RAID to protect all your magnetic disk data.

Disk mirroring involves writing data to two (or more) identical disks. Should one of the mirrored pair of disks fail, your system continues to operate, uninterrupted, using the other disk. If both disks of a mirrored pair fail at the same time, however, you must recover data from backups. For this reason, you should use EBR to perform regular backups even in a mirrored environment.

RAID offers another method of data protection. A RAID environment typically consists of hardware that provides data redundancy across an array of disk devices. Even if you employ RAID technology, you should also perform regular backups of your data.

For detailed information about disk mirroring and RAID, contact your server hardware vendor and operating system vendor.

Protecting Data through Regular Backups

Regular backups are a common and important way of protecting data on magnetic disk. In a FileNet system, data is initially, and sometimes permanently, stored on magnetic disk. The majority of FileNet systems also include optical storage media to which magnetic disk data is periodically migrated. While on magnetic disk, data is vulnerable to loss through disk failures or “crashes.” If a crash occurs, you must restore data from backups once the failing disk is repaired or replaced.

EBR automates the backup and restore process. You can use EBR to perform your own restores. Performing your own restores saves time

and gives you more control over your environment when situations require immediate attention.

Note EBR does not support backing up or restoring cache objects on a remote BES cache server.

Backing Up MSAR Systems

On MSAR (Magnetic Storage and Retrieval) systems, all EBR backups must be accompanied by concurrent backups of the MSAR surfaces to keep the databases and surface files synchronized.

Note Use your preferred third-party backup method to back up MSAR surfaces; EBR does not support backing up MSAR surfaces.

Full and Interval Backup

You only need to back up the MSAR surfaces that have changed since the last full backup. For example, if you're doing a full EBR backup, you should do a full MSAR backup. Likewise, if you're doing an interval EBR backup, you should do an interval MSAR backup.

MSAR Backup Mode

Online backups **must** be done while the MSAR library is in backup mode, and the MSAR surface files must be backed up while the EBR online backup is running.

If you back up MSAR surface files offline, put MSAR files into backup mode before shutting down Image Services.

Doing this ensures the MSAR surface files include correct checksums and can always be inserted back into the library or imported to a foreign system without problem.

Important The EBR backup that archives MKF data and Oracle data must be run while the MSAR library is in backup mode.

Checksumming

The MSAR library must be in backup mode when backing up the MSAR surface files because of the surface checksumming feature.

Surface checksumming provides protection when a surface is reinserted into the library. However, the checksum for an MSAR surface file is only updated when the surface is ejected or when the MSAR library is put into backup mode. The backup mode process writes checksums to the MSAR surface files, so the files will have current, valid checksums before they are backed up.

Platform Support

EBR is supported for the following operating systems (platforms):

- AIX®/6000
- HP-UX® (HP 9000 and HP Integrity)
- Solaris®
- Windows Server

RDBMS Support and Limitations

- EBR supports Oracle databases, but only those with block sizes of **2 KB**. This will not affect FileNet-controlled Oracle instances, but it might affect site-controlled Oracle instances.
- EBR only backs up Oracle databases used by FileNet Image Services and WorkFlo Queue Services.
- On UNIX servers, EBR only supports raw partitions. (“Cooked” file format is not supported.)
- EBR is supported for disaster recovery of site-provided Oracle in a FileNet-controlled environment, assuming FileNet tools created and continue to manage the databases.
- EBR does not support Oracle’s auto extend function. (EBR backup will be successful, but EBR restore will fail.)
- EBR does not support IBM DB2 or Microsoft® SQL Server.

Tape Library Support

EBR supports Exabyte-Tandberg Data tape libraries. The following table describes the supported Exabyte tape libraries by platform. Note that tape library support is not available on all platforms.

Operating System	Exabyte 210 (8mm)	(Discontinued) Exabyte 218 (4mm)	Exabyte 220 “Mammoth” (8mm)
AIX/6000	Yes	Yes	Yes
HP-UX	No	Yes	No
Solaris	No	No	No
Windows Server	No	No	No

Download the Exabyte library management kit for the tape library driver required for your platform from the Exabyte-Tandberg Data Web site, **www.exabyte.com**. Follow the Exabyte instructions for installing your driver.

The table below lists the FileNet-required driver versions by platform and the FileNet-required driver firmware level by type of tape library. The required driver versions, later driver versions that are installed from the Exabyte Web site, the required firmware level, and later firmware levels are all supported. The table also describes the error and warning messages that are received in your system log when you use an earlier or later driver version than the required version. We suggest you follow the recommended action in response to either message.

Exabyte Tape Library Driver Information

Platform	Required Driver Version	Required Firmware Level	if Tape Lib Type is:	If Driver Version Is Earlier than Required Version	If Driver Version Is Later than Required Version
AIX/6000	Version 1.0 or later	4.13.5 or later	EXB-210	**Error Message:** Unsupported tape library driver version ***Recommended Action:*** The operation is aborted because earlier driver versions are not supported. Download the latest version from the Exabyte Web site.	***Warning Message:*** Tape library driver <CHGR version 1.x> by Exabyte Corp. has not been certified by FileNet. ***Recommended Action:*** Later driver versions may be used. Report any problems to your service representative.
HP-UX	Version 1.3 or later	4.13.5 or later	EXB-218		
		4.11.8 or later	EXB-220		

To determine which tape library driver version has been installed, use the TLIB_tool's inventory command to display information about the tape library, including the tape library driver name and driver version number. See **"TLIB_tool" on page 351** for more information.

To determine the current firmware installed on your tape library, look on the library front panel.

Tape Support

EBR supports several tape device types. However, not all platforms support all types. The following matrix describes the supported tape devices by platform:

Operating System	Standard 8mm	Exabyte Model 8900 8mm ("Mammoth")	QIC 1/4" Cartridge	DAT 4mm	Quantum DLT 4000/7000
AIX/6000	Yes	Yes	Yes	Yes	Yes
HP-UX	No	No	No	Yes	Yes
Solaris	Yes	Yes	No	Yes	Yes*
Windows Server	Yes	Yes	Yes	Yes	Yes

*DLT 4000/7000 is configurable on Solaris platforms but has not been tested.

For complete information on hardware compatibility, refer to the *Hardware Compatibility/Dependency Matrix* on the IBM Information Management support page **(www.ibm.com/software/data/support)**.

Heterogeneous Platform Support for Shared Tape Drives

EBR also supports shared tape drives across different operating system platforms, called "heterogeneous platform support." For example, if your environment consists of Sun servers running the Solaris Operating System and PC servers running the Windows Server operating system, you can perform a backup of your Windows Server system using one or more tape drives on the Sun servers. Whether the shared tape drives are in the same domain or different domains is

immaterial to EBR. Heterogeneous platform support improves the throughput rate of the backup by using your available tape drives to the best advantage or by using higher-speed or higher-capacity tape drives that may be available on another platform.

You can mix different tape formats for heterogeneous backup. However, you must use the same platform and same drive type for a restore operation as the platform and drive type used for the backup. For example, if a backup tape was made on an AIX/6000 platform using a high-density 8mm tape drive, you must use an AIX/6000 platform with a compatible tape drive to restore that backup tape. For best results, use the same host and the same tape drive for both backup and restore.

Note Heterogeneous platform support does not include support for backing up data on one type of platform (such as a Solaris system running on a Sun server) and restoring that data on another platform type (such as a Windows Server system running on a PC server).

Heterogeneous Platform Support Example

The backup script example below performs a full, offline backup of the index database on the system named “coyote” using heterogeneous platform support for shared tape drives. The script calls out two ‘include’ files: datasets.inc, to define all datasets and an include file and coyote.dev, to define the backup device. The index database on “coyote” is striped into two parts. Thread 1 uses the local 8mm tape drive on “coyote,” while thread 2 uses the 4mm tape drive on the remote system named “rojo,” which is accessed over a network connection. For a diagram of heterogeneous tape support across platforms, see **“System Hardware Configuration for Heterogeneous Tape Support” on page 39**.

Tip All the fields in the following EBR script will be described in detail in this manual. Do not try to immediately learn the script if this is your first time reading this manual.

Backup Script for Heterogeneous Tape Support

```
EBR_script ( format_level = 2; );

BACKUP_GLOBAL_PARAMETERS
  volume_group = COYOTE;
  expiration = 30 days;
tape_mount_timeout = 300;
END_BACKUP_GLOBAL_PARAMETERS

#include "/tmp/EBR/datasets.inc"
#include "/tmp/EBR/coyote.dev"

BACKUP_OPTIONS

  Oracle_coyote : backup_options
    full_backup;
    offline_backup;
    archive_redo_log_retention = 14 day;
  end_backup_options

  PermDB_coyote : backup_options
    full_backup;
    offline_backup;
  end_backup_options

  SecDB_coyote : backup_options
    full_backup;
    offline_backup;
  end_backup_options
```

```
  TranDB_coyote : backup_options
    full_backup;
    offline_backup;
  end_backup_options

  PageCache_coyote : backup_options
    full_backup;
  end_backup_options

END_BACKUP_OPTIONS

THREADS

  num_threads = 2;

  thread 1
    device = TAPE_DEV1;
    volume serial = TAPE_SER1;
    datasets
     Oracle_coyote (part 1 of 1);
  end_thread   --thread 1--

  thread 2
    device = TAPE_DEV2;
    volume serial = TAPE_SER2;
    datasets
     PermDB_coyote  (part 1 of 1),
     SecDB_coyote  (part1 of 1),
     TranDB_coyote  (part 1 of 1),
     PageCache_coyote  (part 1 of 1);
  end_thread  --thread 2--

END_THREADS
```

Datasets Definition File (tmp/EBR/datasets.inc) for Heterogeneous Tape Support

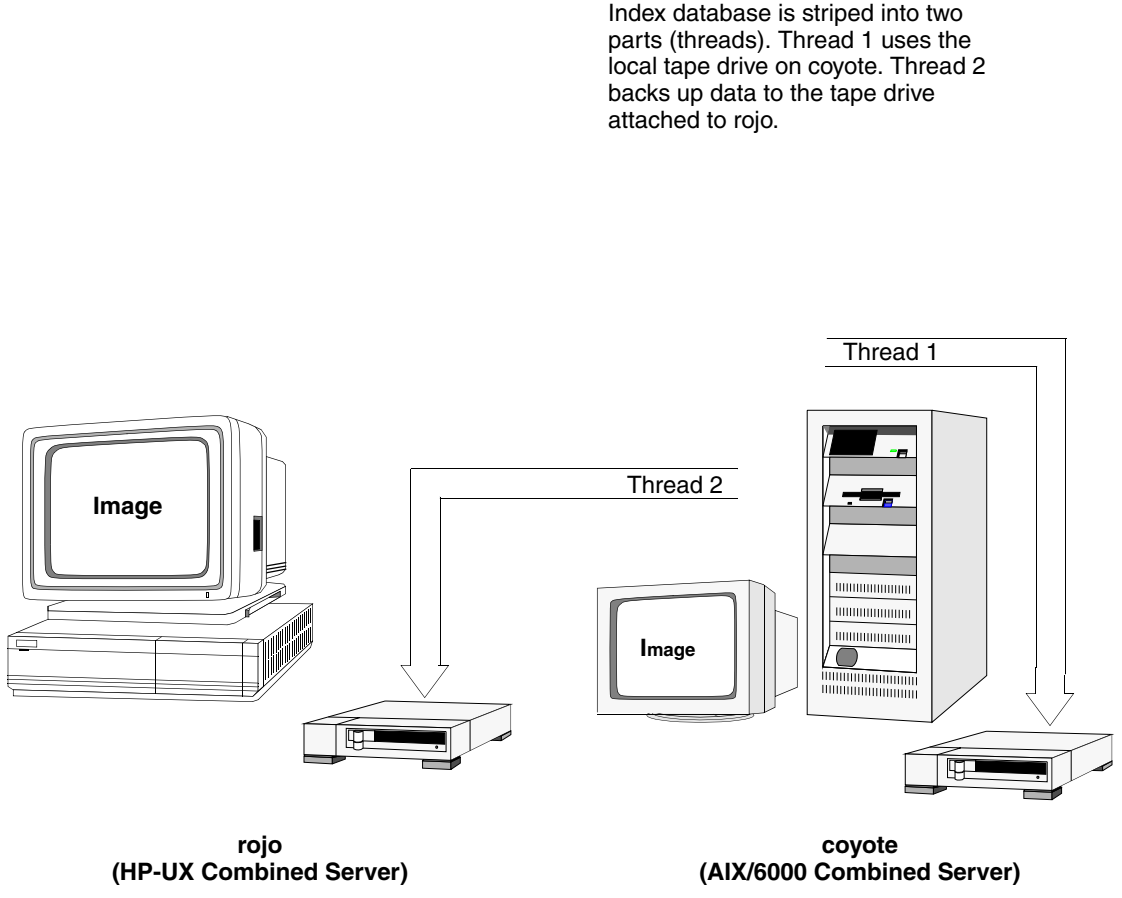
```
DATASETS
```

```
------------------- Domain           coyote:FileNet -------------------
--  coyote              AIX    : Server ID: 1, Server Type : Combined
-----------------------------------------------------------------------
-- Begin Datasets for Server 'coyote' of 'coyote:FileNet' -------------

  Oracle_coyote : Oracle
--  DatasetSize = 356,515,840
    location = "coyote";
    signature_file_directory = "/fnsw/local/oracle/ora_sig";
  end_Oracle

  PermDB_coyote : MKF
--  DatasetSize = 146,800,640
    location = "coyote";
    base_data_file = "/fnsw/dev/1/permanent_db0";
  end_MKF

  SecDB_coyote : MKF
--  DatasetSize = 16,777,216
    location = "coyote";
    base_data_file = "/fnsw/dev/1/sec_db0";
  end_MKF

  TranDB_coyote : MKF
--  DatasetSize = 62,914,560
    location = "coyote";
    base_data_file = "/fnsw/dev/1/transient_db0";
    transient_db;
  end_MKF

PageCache_coyote : cache
--  DatasetSize = 83,886,080
    location = "coyote";
    transient_db = TranDB_coyote;
    permanent_db = PermDB_coyote;
    security_db = SecDB_coyote;
```

```
  end_cache
-- End Datasets for Server 'coyote' of 'coyote:FileNet' -------------------
-------------------- Order Constraints --------------------

  order_constraints
    SecDB_coyote, PermDB_coyote, TranDB_coyote before PageCache_coyote;
  end_order_constraints

END_DATASETS
------------------------------------------------------------------
-- This file contains datasets for the following domains:
coyote:FileNet

Device Definition File (tmp/EBR/coyote.dev) for Heterogeneous Tape Support
```

```
DEVICE SPECIFICATIONS
 TAPE_DEV1: tape_drive                                -- required if backup
device is tape drive
   location = "coyote";                               -- required
   drive_name_rewind = "/dev/rmt0";                   -- required
   drive_name_no_rewind = "/dev/rmt0.1";              -- required
   drive_type = 8mm;                                  -- required
 end_tape_drive

 TAPE_DEV2: tape_drive                                -- required if backup
device is tape drive
   location = "rojo";                                 -- required
   drive_name_rewind = "/dev/rmt/0m";                 -- required
   drive_name_no_rewind = "/dev/rmt/0mn";             -- required
   drive_type = 4mm;                                  -- required
 end_tape_drive

 END_DEVICE_SPECIFICATIONS
```

System Hardware Configuration for Heterogeneous Tape Support



Basic Procedures

EBR backup and restore procedures consist of a basic set of steps. Although you will likely develop procedures unique to your environment, the following sections represent general steps you must take to back up or restore your FileNet system. Included are references to chapters that contain details for performing each step.

Basic Backup Procedures

Step	Reference
Ensure that Oracle archive log mode is enabled if you plan to perform online backups of an Oracle index database.	See **“Appendix F – Enabling Archive Log Mode” on page 377**.
Select the method of backup – offline, online, full, interval. If you plan to perform online backup of an Oracle database, archive log mode must be enabled.	See **Chapter 3, “Quick Start,” on page 73** and **Chapter 4, “Developing Your Backup Strategy,” on page 89**.
Create an Oracle signature file directory.	See **“Oracle Databases” on page 192**.
Develop your backup and restore scripts.	See **Chapter 5, “Developing Your Scripts,” on page 147**.
Prelabel your backup tapes.	See **“EBR_label” on page 332**.
Prepare your system for the type of backup you are performing.	See **Chapter 4, “Developing Your Backup Strategy,” on page 89** and **Chapter 6, “Running The Backup Script,” on page 237**.
Back up your system.	See **Chapter 6, “Running The Backup Script,” on page 237**.
Store your backup tapes.	See **“Organizing Your Backup Tapes” on page 146**.

Basic Restore Procedures

Step	Reference
Collect all backup tapes to be used in the restore.	See “Preparing for Restore” on page 249.
Prepare your system for the restore operation.	See “Preparing for Restore” on page 249.
Restore your system.	See “Restore Procedure” on page 253.
Back up your system when the restore operation completes.	See Chapter 6, “Running The Backup Script,” on page 237 and Step 8 on page 257.

2
Understanding EBR Concepts

To understand EBR and how to use it, you should first become familiar with terms and concepts used throughout this guide. This chapter:

- Defines commonly-used terms, giving examples where appropriate
- Presents EBR concepts
- Describes how EBR implements its concepts through scripts and programs

Commonly-Used Terms

This section defines terms you see frequently when dealing with backup, restore, and EBR.

Media

Media is any material on which data is stored (magnetic disk, optical storage disk, magnetic tape). The term **storage media** generally refers to optical storage disks and MSAR surfaces. For more information about Magnetic Storage and Retrieval, see *MSAR Procedures and Guidelines*. To download IBM FileNet documentation from the IBM support page, see **Accessing IBM FileNet Documentation**.

Backup

A **backup** refers to copying data from magnetic disk to other media, usually magnetic tape. You have several types of backup to consider when developing your backup and restore strategy.

Full and Interval Backup

You can perform a **full backup** or an **interval backup**. A full backup copies all the data you specify to tape or magnetic disk. An interval backup copies only the data that has been modified since the last full backup.

You **must** perform one full backup before you perform the first interval backup. Because an interval backup is smaller, restoring an interval backup is several times faster than restoring the full backup. Interval backups are cumulative since the last full backup.

For example, you may choose to perform a full backup each weekend, and interval backups each night during the week. As the week progresses, each interval backup contains all the data contained in the previous day's interval backup plus the additional data that was modified that day. The most recent interval backup contains all the most recently modified data. Therefore, you will have only one current interval backup tape at any time.

If you ever need to restore the data, you'll only need to restore the full backup and the most recent interval backup.

Offline and Online Mode

Offline backup is performed with the FileNet databases and Image Services software inactive (shut down). Online backup is performed

with the databases and Image Services software active, with users performing normal work as the backup occurs.

EBR allows for both full and interval backups in **offline** and **online** mode. The only exception is that interval backups can't include cache or the transient database.

For example, you may choose to perform a full offline backup each weekend when your system is idle, and interval online backups each night during the week when the system is busy.

Note EBR can only perform full offline backups of disk cache. EBR backs up only **locked objects** and only those locked objects that are **not** temporary. (Locked objects are those that cannot be deleted from cache, including committed documents that are not yet permanently stored on magnetic disk or storage media. See **"Backup Constraints for Cache and the Transient Database" on page 124** for a description of temporary objects.) In addition, cache backups must be accompanied by a full offline backup of the transient database. Unlike a database, a cache cannot be rolled forward (updated to the most current transaction). Therefore, two protection techniques for cache are FileNet **CSM_exim** utility and disk mirroring.

The following table identifies valid backup types by dataset:

Backup Modes by Dataset

Dataset	Dataset Type	Full Backup	Interval Backup	Online Backup	Offline Backup
Security database	MKF	Yes	Yes	Yes	Yes
Permanent database	MKF	Yes	Yes	Yes	Yes

Backup Modes by Dataset, Continued

Dataset	Dataset Type	Full Backup	Interval Backup	Online Backup	Offline Backup
Transient database	MKF	Yes	No	No	Yes
Cache	Raw partition (UNIX) or file (Windows Server)	Yes	No	No	Yes
Index database	Oracle	Yes	Yes	Yes	Yes

For more information on each dataset type, see the table **"Overview of FileNet Datasets and Cache" on page 91**.

Immediate and Delayed Backup

You can schedule an EBR backup as an **immediate backup** or a **delayed backup**. An immediate backup occurs as soon as you submit your backup script. A delayed backup occurs at some specified future time. Delayed backups are also called scheduled backups or deferred backups. See **"Immediate versus Delayed Backups" on page 114** for more information.

Unattended Backup

Most delayed backups are run in **unattended** mode. The backup tape is preloaded in the tape drive and no operator is present during the backup.

Important On Magnetic Storage and Retrieval (MSAR) systems, the MSAR libraries **must** be put into backup mode before the EBR backup starts.

To do this, you can create a script to start DOC_tool and set the mode. The MSAR surfaces **must** be backed up at the same time as the EBR backup to keep the databases and surface files synchronized.

Use your preferred third-party backup method to backup MSAR surfaces; EBR does not support backing up MSAR surfaces.

To schedule an unattended backup, use the **crontab utility** in a UNIX environment or the **AT command** in a Windows Server environment.

Refer to **"Unattended Backups" on page 120** for more information on running unattended backups.

Restore

A **restore** refers to copying good data from backup media (usually a magnetic tape) to your system magnetic disks. Restores are done infrequently, usually after a disk crash or a disaster has destroyed or corrupted data on magnetic disk.

During a restore operation, EBR automatically **rolls forward,** or reconstructs**,** a database to the end of the last completed transaction using the **recovery log** resident on magnetic disk at the time of the restore. No operator intervention is necessary.

The recovery log contains entries for all changes made to a database since the last backup of the database completed. FileNet Multi-Keyed File (MKF) software writes to the MKF recovery log in a circular manner. When the log is full, database changes are recorded in the next sequential log, if available. MKF recovery log files are not automatically archived (closed and written to tape or magnetic disk) when full. When the last log file is full, database changes are written to the first log file—**overwriting** data on the first log.

To prevent overwriting of log data, follow these recommendations:

- Allocate sufficient disk space for MKF recovery logs to prevent overwriting between backups.

  Sufficient disk space means a magnetic disk allocation large enough to hold two times the amount of data normally generated between backups.

- Back up the databases at regular intervals.

A restore of an MKF database automatically attempts to apply (splice in) recovery log data to the database. Splicing in recovery log data brings the MKF database forward in time to the end of the last complete transaction. Frequent backups minimize the amount of data that must be kept in recovery logs.

EBR uses RDBMS recovery logs (called “redo logs” in Oracle) to roll the database forward during database recovery.

At the completion of the restore of a full backup tape, the rollforward operation applies recovery log data to the database to ensure that the database is restored to the most current state possible and without losing work done since the last backup. If an interval restore is to follow the full restore, EBR does not permit rollforward to occur until **after** the interval backup tape is completely restored. After the interval restore completes, automatic rollforward recovery occurs.

Scripts

A **script** is an ASCII file that contains a set of instructions to perform tasks. EBR is script-based: you use scripts to capture and enforce your enterprise backup and restore policy. You define a backup and restore

strategy for magnetic disk datasets that are resident on the FileNet servers in your enterprise. Then you capture that strategy in backup and restore scripts.

Using scripts guarantees that backup and restore strategy is followed exactly. The System Administrator makes all decisions in advance and records those decisions in the scripts. EBR enforces the decisions.

Sample scripts are available with the EBR product. Some of the sample scripts are for use with the Quick Start approach described in **Chapter 3, "Quick Start," on page 73**. You can use other samples as a starting point for writing your own customized scripts. See **Chapter 5, "Developing Your Scripts," on page 147** for details of how to develop customized scripts and for examples of both sample and customized scripts.

You can also develop your own customized scripts with the EBR_genscript tool. After starting the tool, you interactively respond to prompts. EBR_genscript uses your responses to generate a datasets definitions file, a backup script, or a restore script. See **"EBR_genscript" on page 268** for more information.

High-Performance Techniques

EBR provides several high performance features:

- Threads
- Pipelining
- Parallel processing of striped datasets
- Data compression

These techniques, used separately or together, streamline the backup and restore process. When used together, the effects of each are multiplied by the others, up to the physical limits imposed by the hardware.

Threads

An EBR **thread** is a single serial stream of data to or from a tape cartridge. A thread has a one-to-one association with a tape cartridge. EBR uses threads to enable backup operations to be performed concurrently to multiple tape drives, greatly increasing performance. Similarly, EBR uses threads to perform restore operations concurrently from multiple tape drives. Threads of a backup or restore process, which generally operate concurrently, operate sequentially if multiple threads share a tape drive. You specify the detailed characteristics of each concurrent activity in the THREADS section of your script. The total number of threads is limited by the amount of available server memory. (See **“Appendix G – Memory Requirements” on page 384**.)

You can back up or restore different datasets concurrently provided that the datasets do not use the same tape drive and they do not have interdependencies. For example, a concurrent backup of cache and the transient database is not possible. Since cache backup is dependent on the successful completion of the transient database backup, the cache backup waits.

You can break large datasets into multiple parts. EBR assigns these dataset parts to different threads for processing. If you have multiple tape drives, the dataset can be backed up faster. (See **“Parallel Processing of Striped Datasets” on page 52**.) If a dataset is too large to fit onto one tape cartridge, you **must** break it up into multiple parts so each part will be assigned to a different cartridge.

All cartridges involved in a backup must be in the same **volume group**. A volume group is a named collection of backup tape cartridges. EBR requires **exactly one thread per tape cartridge**. The number of tape cartridges may be greater than the number of tape drives in your system. If so, some threads must share some tape drives. Tape drives are in use concurrently during an EBR backup or restore.

If the number of tape drives is equal to the number of tape cartridges, threads start and run concurrently on their associated tape drives.

If the number of tape drives is less than the number of tape cartridges, the threads gain access to the tape drive in increasing thread number order. For example, if threads 1, 2, and 3 share the same tape drive, thread 1 executes first, followed by thread 2, and then thread 3. Benefits of this approach include:

- Backups and restores can be performed even if multiple tape drives are unavailable. In fact, all tape cartridges can share a single tape drive.
- Overflow tapes, which are not supported in EBR, can be approximated through dataset striping.

  An “overflow” condition occurs when data being written to a tape exceeds the tape’s capacity and the end of the tape is reached. Upon encountering an overflow condition, some applications cause the system to prompt for an additional tape (an **overflow tape**) to be loaded, a process sometimes called “tape switching,” so writing can continue.

  EBR does **not** support cartridge overflow—all backup data must fit on a single tape cartridge. If an overflow condition occurs, EBR aborts and issues an error message indicating it has reached the end of the tape media. You can avoid overflow failures by breaking

up a dataset into multiple parts (dataset striping) and directing the parts to multiple tape cartridges. For more information, see **"Data Overflow" on page 129**.

- A backup performed using multiple threads can be restored concurrently if more tape drives become available in the future.

Pipelining

EBR uses **pipelining** extensively to increase performance.

Pipelining is a technique to overlap CPU and I/O activity when a large number of work units must be performed, the work consists of a fairly small number of fixed sequential steps, and a separate hardware resource exists to process each step.

Each step of the work to be performed is connected in a linear sequence called a pipeline. The output of each step is input to the next step. Since each step runs concurrently on its own hardware resource, the result is an increase in throughput.

The EBR pipeline stages for backup are:

- Reading from magnetic disk
- Compressing the data
- Transmitting data over the network
- Generating error checking and correction (ECC) data
- Writing to tape

Parallel Processing of Striped Datasets

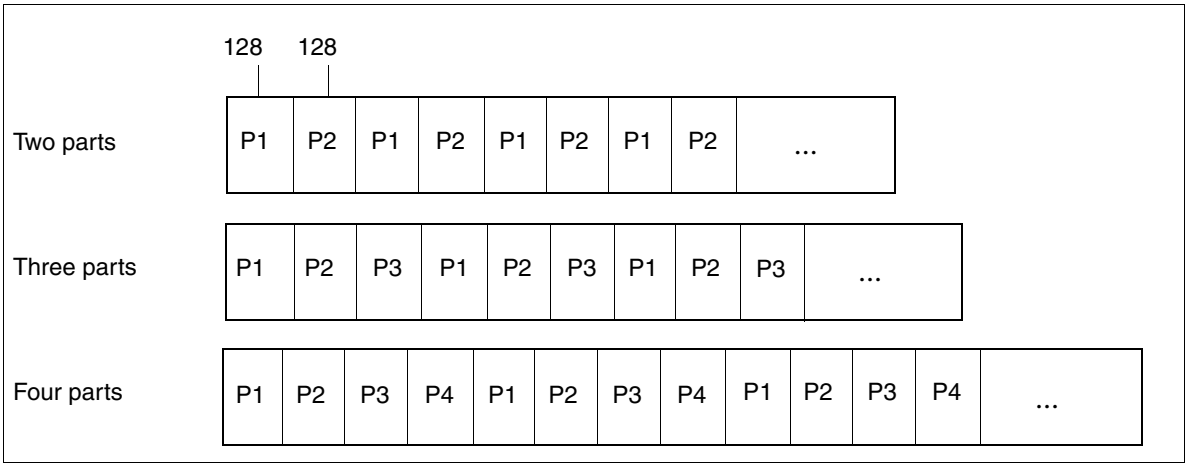
Certain databases in the FileNet system, particularly the index database, can be very large. A sequential backup of such a large database can take considerable time. EBR uses the striping technique to break a single large dataset into multiple smaller parts, called stripes. EBR supports **parallel processing** of multi-part (striped) datasets to reduce the time required to back up large datasets.

Note Small datasets (that is, datasets that can fit on a single tape cartridge) do not need to be striped and generally should **not** be striped. Doing so is inefficient and unnecessary. Specify only one thread for small, unstriped datasets.

However, if your dataset is too large to fit on one tape, you **must** stripe the dataset because EBR does not support data overflow onto multiple tapes. (See **"Data Overflow" on page 129**.)

EBR uses the parameters you specify in the backup script THREADS section to determine the number of parts. Based on the number of parts you specify for a particular dataset, EBR divides the dataset into an equal number of 128-KB segments called chunks. Striping has no effect on database organization or dataset handling outside of EBR.

The following diagram shows the result of breaking a large dataset into two, three, and four 128-KB parts:



Breaking a Dataset into Multiple Parts

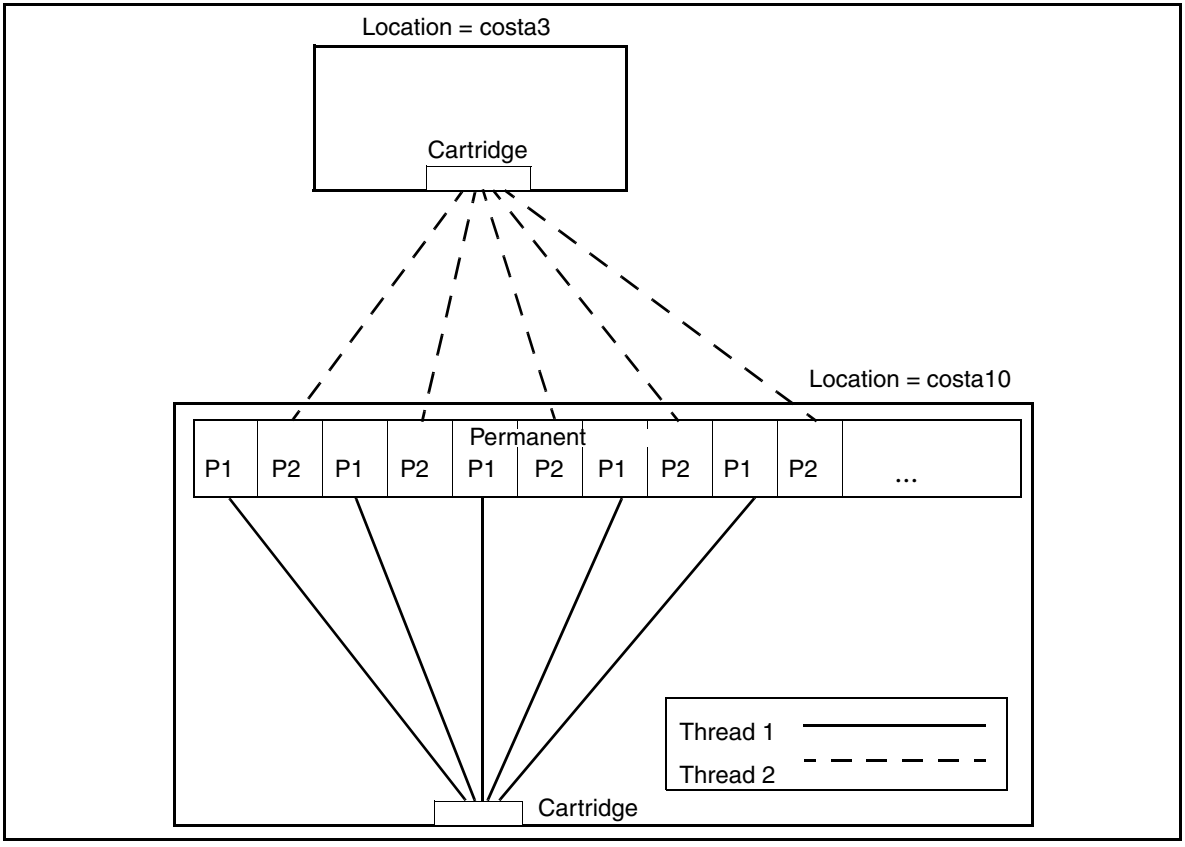
Threads run in parallel when you use multiple tape drives during backup. When threads share a tape drive, they gain access to the drive in increasing thread number order.

Using multiple tape drives concurrently, you can back up each stripe to a separate tape drive, reducing the time to back up the dataset in direct proportion to the number of tape drives available for parallel use. For example, backing up a single large dataset to two tape drives in parallel reduces the elapsed time of the backup by approximately half. If you have only one available tape drive and you have one or more large magnetic disk datasets, consider purchasing more tape drives so that you can run the threads concurrently.

Striping Example with Two Threads

The following diagram represents the permanent database containing a large dataset striped for backup by 2 threads. The permanent database resides on a server named costa10. Each thread backs up the

128-KB segments (P1 or P2) of data belonging to it. In this example, each thread backs up its data to a separate cartridge on a separate tape drive on a server. The location parameter in the backup script THREADS section identifies the name of the server on which a tape drive exists.

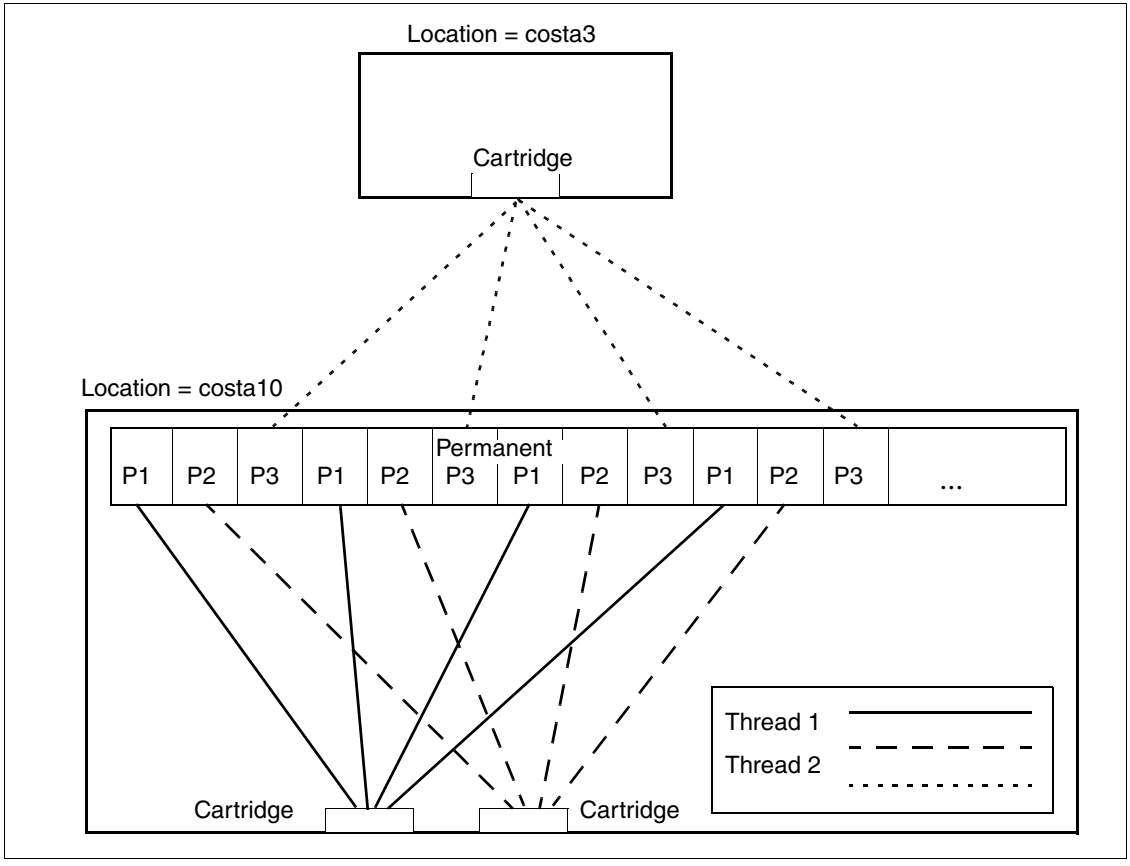


Striping with Two Threads

In the diagram above, thread 1 backs up all P1 segments to a cartridge in a tape drive on the costa10 server. Thread 2 backs up all P2 segments to a cartridge on the costa3 server, which also has a single tape drive.

Striping Example with Three Threads

The diagram below represents the permanent database containing a single large dataset striped for backup by three threads. The permanent database resides on a server named costa10. Each thread backs up the 128-KB chunks of data (P1, P2, or P3) belonging to it. As in the previous example, each thread backs up its data to a separate cartridge on a separate tape drive on a server. The location parameter in the backup script DEVICE SPECIFICATION section identifies the name of the server on which a tape drive exists.



Striping with Three Threads

In this diagram, a second tape drive is available on costa10. Thread 1 backs up all P1 chunks to a cartridge in one tape drive on the costa10 server. Thread 2 backs up all P2 chunks to a separate cartridge in costa10's second tape drive. Thread 3 backs up all P3 chunks to a cartridge on the costa3 server, which has a single tape drive.

The amount of available memory in your system limits the number of threads you can specify for backup. For more information on memory requirements for threads, see **"Appendix G – Memory Requirements" on page 384**.

Note For more information about dataset striping and how to implement striping in your backup script, see **"Special Considerations for Listing Striped Datasets" on page 211**.

Script Example - Striping with Three Threads

The partial script to perform the striping for the three-threaded striping example is shown below. Part 1 and part 2 are sent to separate threads on the costa10 server, which has two tape drives. Part 3 is sent to a separate thread on the costa3 server, which has a single tape drive.

```
--Partial script demonstrating a dataset with three threads.
DATASETS

    perm: MKF
     location = "costa10";           -- hostname of permanent database
     base_data_file = "/fnsw/dev/1/permanent_db0";
    end_MKF

END_DATASETS

THREADS
.
.
 num_threads = 3;
  thread 1
```

```
    device = TAPE_DEV1;
    volume_serial = R1;              -- tape serial number
    datasets
        perm (part 1 of 3);
  end_thread

  thread 2
    device = TAPE_DEV2;
    volume_serial = R2;              -- tape serial number
    datasets
        perm (part 2 of 3);
  end_thread

  thread 3
    device =  TAPE_DEV3;
    volume_serial = R3;              -- tape serial number
    datasets
        perm (part 3 of 3);
  end_thread
END_THREADS
```

Data Compression

When considering backup performance issues, the speed of tape operations (not the speed of magnetic disk accesses) is typically the bottleneck. To alleviate this bottleneck, EBR always compresses backed up data and sends only compressed data to locally-attached tape drives or over the network to remotely-attached tape drives.

Backup and restore of compressed data takes less time than does the backup and restore of uncompressed data. If the compression ratio is 3 to 1, backup time is reduced to one-third of the time required to back up uncompressed data.

The compression ratio EBR uses depends on the type of data you are backing up. The following table shows approximate ratios by data type:

Data Type	Data Compression Ratio
Cache	1:1 up to 1:1.25 Data in cache is already compressed **and in some cases may expand** due to error checking.
MKF data	At least 3:1
Index data	At least 3:1

See **"Compression Factors" on page 138** for compression factors achieved for different types of data.

Additional data compression may be available by enabling a tape drive hardware feature. However, you should you turn off hardware compression for your tape drives for the following reasons:

- EBR automatically compresses the data as it is written so very little additional compression is achieved by the tape hardware compression feature.
- Hardware compression can compromise EBR's automatic error correction capability. For example, EBR always backs up error correction information to log files on the backup tape. The hardware compression feature compresses a number of EBR tape blocks into a single physical tape block. If that single block becomes corrupted, multiple EBR tape blocks are lost. Loss of error correction data in the log files of the lost tape blocks impacts EBR's automatic error correction capability. Turning off hardware compression maximizes EBR's chance of automatically correcting errors.

In general, configure all your tape drives the same way so that any drive of the same type can read or write any tape. If you decide to use

the hardware compression feature of your tape device, enable hardware compression for all drives; if you decide not to use hardware compression, disable the feature for all drives.

Refer to your tape drive hardware manuals and operating system manuals for more information on hardware compression and platform-dependent methods to turn off hardware compression for your tape drives.

Prelabeling Tapes

You must prelabel all tapes you plan to use in EBR backup and restore operations. To prelabel a tape, run the EBR_label utility to assign a serial number and volume group to each individual tape. (See **"EBR_label" on page 332**.) You only need to prelabel a tape once in its lifetime. Prelabeling tapes guarantees that:

- EBR always uses the correct tape according to the backup strategy. (See **"Automatic Tape Recognition" on page 61**.)
- A backup tape cannot be accidentally overwritten before its specified expiration date (EBR refuses to overwrite a tape that has not reached its expiration date and time).

EBR maintains a backup count in the label. You can use the backup count to determine when to retire the tape (for example, before it significantly degrades due to wear). To determine the backup count, run EBR_tdir then examine the "number_of_backups=" field in the output. See **"EBR_tdir" on page 345** for additional information.

Specify the same tape device type in both the EBR_label command and the backup script that uses the tape. Failing to do so may result in

an inability to restore the tape. For guidelines, see **"Selecting Tape Device Type" on page 132**.

Caution A common mistake in the backup process is to insert a tape with the wrong label into the tape drive. This mistake will lock up the datasets. To unlock the datasets, you must use the the following utilities: **"EBR_orreset" on page 344** and **"EBR_ulmk" on page 351**.

Automatic Tape Recognition

Once a tape cartridge has been labeled, an operator inserts the tape cartridge into, or removes it from, a tape drive. No other operator interaction is necessary. EBR recognizes when a tape is in the drive, reads the tape label information, and automatically recognizes the correct tape. If the wrong tape in the drive, EBR rejects the tape.

Note EBR never requests a scratch tape (a tape that has not been labeled for backup purposes).

Byte Order Independence

Two types of central processing unit (CPU) **byte ordering** exists—**big-endian** and **little-endian**. Byte ordering is the sequence in which the CPU processes bytes of data and sends them to an I/O device (a tape drive, for example). Big-endian CPUs process bytes starting from the big end of a four-byte longword whereas little-endian CPUs process the bytes of the longword starting from the little end.

Big-endian CPUs include the IBM RS6000, IBM Power PC, HP 9000, SUN SPARC, and Motorola MC 68040.

Little-endian CPUs include Intel 80486, Pentium, and DEC VAX.

Byte order becomes important when you consider different scenarios that may occur in an enterprise. For example, if you are backing up data across a network, the byte order of the host system owning the tape drive may be different from the byte order of the system containing the disk datasets. In another example, the byte order of the host system owning a tape drive when a restore is performed may not be the same as the byte order of the system that owned the tape drive when the backup was made. Transferring data between systems of different byte order can result in corrupted data if the byte order differences are not resolved during the transfer.

Fortunately, EBR takes care of most byte ordering issues for you. EBR is independent of the byte ordering of the system to which the tape drive is attached, allowing you to back up to and restore from systems of different byte order than the systems containing the disk data. EBR automatically recognizes the proper byte ordering to use in processing data from MKF databases across systems of different byte order, including mixed byte order systems and multiple byte order systems. In fact, you can back up MKF databases on one system and restore them to a system of different byte order that also uses different file names and different file sizes for the databases. (However, the logical definitions of the datasets should be the same on the two systems.) EBR handles the byte ordering and the file name differences of MKF datasets for you.

For other types of data (for example, data from relational databases), you may have to perform additional conversion when transferring data

between systems of different byte order. For more information on byte ordering, see **“Appendix D – Byte Ordering Issues” on page 371**.

Automatic Error Detection and Correction

EBR employs features to make backups and restores extremely reliable.

To detect I/O errors in backup data, EBR includes a **checksum** in every block of data. A checksum is the arithmetic sum of the binary data in the block. When data is sent across an I/O channel, the checksum calculated on the sending side is also sent. EBR recalculates the checksum on the receiving side of the transmission and compares it to the checksum calculated on the sending side. An unequal comparison indicates the data is corrupted and EBR resends the data. EBR retries the operation up to 1,000 times before it terminates the operation.

EBR always backs up error correction information to tape. When such information is available, a restore can succeed even if multiple tape blocks are unreadable. If data being read from tape is found to be corrupt, EBR uses the error correction information on tape to correct the data before the data is restored to magnetic disk.

EBR’s automatic error correction also eliminates the need for a time-consuming backup tape verification step.

Status Reporting

During normal operation, EBR stores information about RDBMS backup and restore activity into temporary files in its own working directory:

/fnsw/local/tmp/EBR/	For UNIX platforms
<drive>:\fnsw_loc\tmp\EBR\	For Windows Server platforms

Upon normal completion of an operation, EBR deletes the temporary files. However, if the operation does not complete normally, EBR does not delete the files. The contents of these files may help you trouble-shoot and resolve problems.

Note You should not store any other files in EBR's working directory.

EBR reports status in the user interface display and in the following log files:

- Progress log
- Summary log
- System event log

User Interface Display

In the user interface display, you can observe all concurrent activity occurring while EBR is running. The user interface display runs in a shell window which you must establish as described below.

Establishing User Interface Display Windows

EBR uses X windows (for UNIX environments) or DOS windows (for Windows Server environments) to display the progress log and status information. Use your operating system's techniques to open new status display windows as necessary. Windows should be at least 80 columns wide and at least 25 lines tall.

Tip The height of the window depends on the number of threads in your script. If the height of your window is insufficient to display all the thread activity, information scrolls out of the viewable area of the window. To prevent scrolling, increase the number of lines in your window until scrolling no longer occurs.

In AIX/6000 Release 4.1 and later, system error log (syslog) messages are written to the first new X window that you open. In the Windows Server environment, syslog messages for a utility are written to the open window in which the utility is running. In either case, **important EBR status and other messages are overwritten by syslog messages**. Use the following workarounds to avoid overwritten EBR messages in these two environments:

AIX

- For AIX/6000, create a dummy file called **nocons** in the /fnsw/sd directory. Verify the following file is present in your system:

  /fnsw/sd/nocons

WIN

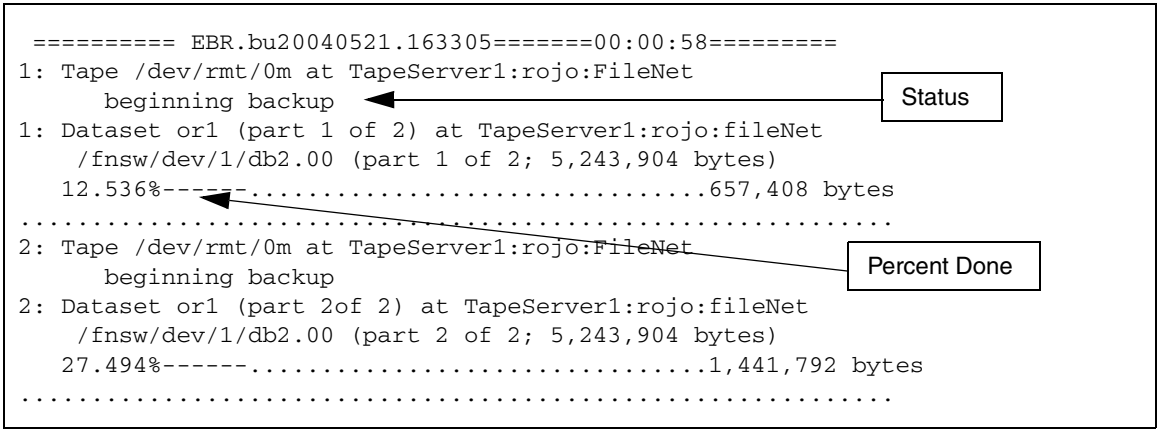
- For Windows Server, create the \fnsw\sd directory if it does not already exist. Create a dummy file called **nocons** in that directory. (You do not need to include a file extension.) Verify the following file is present in your system:

  \fnsw\sd\nocons

User Interface Display Information

The user interface display heading reflects the progress log name that EBR creates when the backup or restore operation starts and the elapsed time of the operation in hh:mm:ss format. EBR indicates required operator actions (such as inserting tape cartridges into drives) with messages in capital letters.

To put the progress of the EBR operation in perspective, you can examine the percentage of work completed and number of bytes processed. However, for operations involving small amounts of data, the operation may complete quickly, before the percentage completed line can display.



User Interface Display

The EBR user interface display is designed for the general case of running multiple threads concurrently. Refer to the example above as you read the following descriptions.

The EBR thread number (exactly one per tape cartridge) displays as a digit in the first column. The same thread number appears twice in column 1. The first line on which it appears and the line below give the status of the tape drive. The digit appears again in column 1 on the third line. The third, fourth, and fifth lines show the status of the magnetic disk dataset being backed up. This format is repeated for each active thread.

For datasets with a known size (for example, a partition or a database), EBR displays total bytes, percentage done, and accumulated bytes in the user interface display. However, for datasets of unknown size (such as a partition with an unspecified size), EBR displays only the accumulated bytes.

Entries in the user interface display may not always appear in the same sequence as specified in the EBR script. The tape drive may be on the same system as the magnetic disk dataset being backed up or on a different system. Since EBR tasks occur concurrently and a significant amount of buffering occurs in main memory, disk processing can get significantly ahead of tape processing. For example, if a dataset is small enough, the user interface display may indicate that the dataset is completely backed up by the disk side before the tape side has even finished opening the tape.

Progress Log

EBR writes a detailed progress record of each backup or restore operation to an ASCII file called a progress log. The progress log resides in the following directories:

/fnsw/local/logs/EBR	For UNIX systems
<drive>:\fnsw_loc\logs\EBR	For Windows Server systems

Note For Windows Server systems, your log files may be located on any magnetic disk drive.

At the completion of a backup or restore, EBR displays the progress log file name. The file name format is one of the following:

EBR.bu*yyyymmdd.hhmmss*

EBR.re*yyyymmdd.hhmmss*

The EBR.bu prefix identifies a backup progress log; the EBR.re prefix identifies a restore progress log. The remainder of the progress log file name is the year, month, day, hour, minute, and second that the backup or restore started.

Use a text editor (such as vi, more, less, emacs, or the Windows Server Notepad) to view the progress log.

The example below is a partial listing of a backup progress log:

```
Enterprise Backup/Restore Progress Log
Wed Sep  3 10:22:13 2003
Processes and their assigned letters:
M = master process. Makes RPC's to all tape rqh. processes (one per thread).
T = tape requesthandler process.  Forks a "t" and an "n" process.
    t = tape worker process.   Performs I/O on a single tape drive.
    n = network (N/W) disk client process.  Makes RPC's to a disk rqh. process.
D = disk requesthandler process.  Forks a "d" and a "c" process.
    d = disk worker process.  Performs I/O on files of a disk dataset.
    c = disk data compressor/decompressor process.

(In the next line, 'N' is thread Number, 'L' is process Letter:)
hh:mm:ss NL   pid    message
-------- -- -------- --------------------------------------------------
10:22:14 0M    30269 Master process:  BEGIN BACKUP
10:22:23 1n    28224 Begin N/W disk client process for backup
10:22:23 1n    28224 Begin backup of dataset part (part 1 of 1)
10:22:23 1t    35137 Begin tape worker process for backup. Backup to disk file
                     "/tmp/ebr/fake_tape" at "TapeServer1:rojo:FileNet" .
10:22:23 1t    35137 NEED TAPE serial=TEST_SER volume_group=TEST_VG IN DRIVE
                     /tmp/ebr/fake_tape AT TapeServer1:rojo:FileNet !
10:22:23 1t    35137 backup disk file opened
10:22:23 1t    35137 Read 1 compressed blocks (32K) cumulative from tape.
(continued on next page)
```

```
(continued from previous page)
...
10:23:10 1t    35137 Ejecting tape  at TapeServer1:rojo:FileNet and writing
                     tape mark.
10:23:10 1t    35137 SUCCESSFUL end backup tape worker process
10:23:10 1n    28224 Got 140 final total compressed blocks (32K) from disk
                     server.
10:23:10 1n    28224 SUCCESSFUL end N/W disk client process for backup
10:23:11 0M    30269 Closing thread 1
10:23:11 0M    30269
BACKUP SUCCESSFULLY COMPLETED
Wed Sep  3 10:23:11 2003
```

Use the progress log to determine the success or failure of each part of the backup or restore operation. Support personnel use the progress log information to diagnose problems. To learn more about EBR's internal processing, you can also examine the progress log entries as you read the information in **"Appendix C – EBR Program Operation" on page 363**.

You can also use the progress log to calculate the throughput of every part of the backup or restore process. You can determine how much data was backed up from disk, how much data was written to tape, and how much data was sent over the network. For more information, see **"Appendix K – Calculating Throughput" on page 425**.

Over time, accumulated progress logs could consume significant amounts of magnetic disk space. To prevent this, EBR automatically deletes progress logs older than 30 days. To keep logs longer than 30 days, copy them to tape before the deletion date occurs.

Summary Log

EBR maintains a short summary of backup and restore operations in an ASCII file called the summary log. EBR writes the summary log to the following directory:

/fnsw/local/logs/EBR	For UNIX systems
<drive>:\fnsw_loc\logs\EBR	For Windows Server systems

Tip For Windows Server systems, your log files may be located on any magnetic disk drive.

The file name of the summary log is:

EBR_summary.log

EBR writes a message to the summary log each time a backup or restore starts, completes successfully, or fails. From this log, you can tell at a glance if the last backup or restore succeeded or failed. The order of the datasets listed in the summary log file follows the order of datasets defined in the DATASETS section of your script. From the dataset list, you can determine the backup history for a dataset.

Use a text editor to view the summary log. A sample is shown below:

```
start backup:                  Tue Jun 15 09:14:45 2004
    volume group = VG
    datasets: part sec perm inxdb
end   backup (SUCCESSFUL):    Tue Jun 15 09:19:36 2004
    volume group = VG
    datasets: part sec perm inxdb
start backup:                  Tue Jun 15 15:38:29 2004
    volume group = VG
    datasets: sec perm inxdb
end   backup (SUCCESSFUL):    Tue Jun 15 15:43:31 2004
    volume group = VG
    datasets: sec perm inxdb
```

If the summary log does not exist, EBR automatically creates one. EBR does not automatically delete an existing summary log or any of its entries. At some point, old summary log information is probably not useful. Use a text editor to delete old entries. If the log file becomes too large, you can archive the file to tape and delete the magnetic disk-resident copy. If you manually delete the summary log without archiving a copy to tape, EBR creates a new summary log but **old information will be lost**.

System Event Log

EBR writes error messages to the system event log of the host that encounters the error. EBR also writes informational messages to the system event log each time a backup or restore operation starts or completes. Image Services stores system event logs in one of the following locations:

/fnsw/local/logs/1/elog*yyyymmdd* For UNIX systems

<drive>:\fnsw_loc\logs\1\el*yyyymmdd* For Windows Server

where *yyyymmdd* is the year, month, and day of the event log.

Tip In Windows Server systems, log files may be in a different directory than the default directory given above. Log files are in the directory you specified during FileNet software installation.

Use the System Monitor to examine the contents of the system event log. See the *System Administrator's Handbook* for more information. To download IBM FileNet documentation from the IBM support page, see **Accessing IBM FileNet Documentation**.

3
Quick Start

Using the Quick Start approach, you can try out the EBR backup program immediately without first planning the backup strategy for your enterprise. After you become familiar with EBR using the Quick Start approach, you will be ready to develop a backup strategy for your unique environment using the information provided in the remainder of this guide.

To use the Quick Start approach, select a script from a set of prewritten backup scripts—for backup to tape or for backup to a magnetic disk file. Modify the script to meet your environment needs, prelabel your tapes or disk file, and run the script. You do not have to shut down your system. The Quick Start scripts perform an online backup of the MKF security database, which is typically very small.

The Quick Start scripts can be found in:

/fnsw/local/EBR/samples	For UNIX systems
<drive>:\fnsw_loc\EBR\samples	For Windows Server systems

Each Quick Start backup method is described in the following topics.

Quick Start Using Tape

Before running Quick Start using tape, select your tape device from the following EBR-supported tape devices:

- Standard 8mm
- Exabyte Model 8900 (“Mammoth”) 8mm
- QIC (quarter-inch cartridge)
- 4mm or DAT

CAUTION Be sure to specify the correct tape device type in both the EBR_label command and your backup script. See **“Selecting Tape Device Type” on page 132** for more information.

Now you are ready to perform some preparatory steps, as described below.

Preparation

To prepare for the backup, perform the following tasks:

1 Select a tape serial number.

For Quick Start, simply use **R1** as the tape serial number.

When performing backups in a production environment, choose a unique tape serial number and a volume group name for the tape. Tape serial numbers should be unique across all tapes in your enterprise.

2 Select a tape volume group.

All tapes involved in a backup must have the same volume group name. For Quick Start, simply use **VG1** as the volume group name.

However, in a production environment, the volume group name should be made up of pertinent information, such as the type of backup, the datasets involved, and the day of the week the backup was made.

Note The syntax for tape serial number and volume group name is restricted. The restrictions are satisfied by the names R1 and VG1. For an explanation of restrictions, see **"EBR_label" on page 332** or the online help for EBR_label.

3 Prelabel your tapes. (See **"EBR_label" on page 332**).

Use EBR_label to prelabel your tapes with the selected tape serial number and volume group. For example, if you selected 8mm tape as your backup device and your tape device name is /dev/rmt0, your EBR_label command is:

EBR_label tape=/dev/rmt0 type=8mm ser=R1 vg=VG1

You can run the EBR_tdir utility to display the tape label you created. (See **"EBR_tdir" on page 345**.)

4 Make a backup copy of FileNet-provided sample scripts.

We recommend that you preserve the original, unmodified sample scripts. Create a directory and copy all the FileNet-provided sample scripts from the sample directory to your new directory.

Sample scripts are located in the following directories:

/fnsw/local/EBR/samples	For UNIX systems

<drive>:\fnsw_loc\EBR\samples For Windows Server systems

5 Select a backup script, quickbu.tap or quickbu.dsk, from your copy of the sample scripts.

6 Edit the backup script as appropriate for your environment.

The quickbu.tap script for a tape backup on an AIX/6000 is shown as follows:

```
EBR_script( format_level = 2; );
BACKUP_GLOBAL_PARAMETERS
     volume_group = VG1;                          -- tape volume group name
     expiration = 1 day;
     tape_mount_timeout = 300;
END_BACKUP_GLOBAL_PARAMETERS

DATASETS
     sec: MKF
       location = "demo";                          -- hostname of security
database
       base_data_file = "/fnsw/dev/1/sec_db0";
     end_MKF
END_DATASETS

DEVICE_SPECIFICATIONS
  TAPE_DEV1 : tape_drive
    location = "demo";
    drive_name_rewind = "/dev/rmt0";
    drive_name_no_rewind = "/dev/rmt0.1";
    drive_type = 8mm;
  end_tape_drive
END_DEVICE_SPECIFICATIONS

(continued on next page)
```

```
(continued from previous page)

BACKUP_OPTIONS
     sec: backup_options
        full_backup;
        online_backup;
     end_backup_options
END_BACKUP_OPTIONS


THREADS
     num_threads = 1;
     thread 1
       device = TAPE_DEV1;
       volume_serial = R1;                    -- tape serial number
       datasets
           sec;
     end_thread
END_THREADS
```

The first line in the script specifies the syntax format level in which the script was written. (For more information, see **"Script Syntax Format Level" on page 157**.)

Main sections follow the script syntax format level specification:

- BACKUP_GLOBAL_ PARAMETERS
- DATASETS
- DEVICE_SPECIFICATIONS
- BACKUP_OPTIONS

- THREADS

BACKUP_GLOBAL_PARAMETERS Section

Global parameters specify the volume group name, an expiration limit, and the tape mount timeout value. The quickbu.tap script specifies that:

- The volume group is VG1.
- The backup tape created by the script may be overwritten after one day.
- The tape mount process will be aborted and an error message returned when a tape error occurs and the tape timeout value (in seconds) is exceeded. A tape error occurs when no tape is inserted or an invalid tape is used. When the error occurs, EBR continues to try to load the tape until the timeout value expires. The default timeout value is no timeout.

DATASETS Section

The security database is the only dataset in the quickbu.tap script. The script assigns the name “sec” to the dataset as a shorthand description. The name “sec” has no special meaning. For purposes of backup and restore, you can choose any names you want for your datasets as long as the names follow the syntax rules for identifiers (see the description of identifiers under **“Tokens” on page 152**).

EBR needs the disk server name or IP address to locate the host on which the magnetic disk dataset resides. You can specify the location name as one of the following:

- Three-part NCH name (for example, “TapeServer1:rojo:FileNet”)

- NCH domain name (for example, “rojo:FileNet”)
- TCP/IP address (for example, “135.0.5.25”)
- Host name (for example, “rojo”)

Host names and IP addresses must be enclosed in double quotes. For more information on location name formats, see **“Specifying Network Host Locations” on page 154**.

When using NCH three-part names or the NCH domain name, an object with this name must conform to the following restrictions:

- The object must be in your NCH database.
- The object must have the network address property.
- The value of the object’s network address property must be the network address at which the specified database resides.

On most systems, the first (least significant) part of the NCH name is usually **TapeServer1**. As an example, “TapeServer1:bandit:FileNet” is the NCH three-part name for a system named bandit in the FileNet enterprise. To see all object names in a system, start the **nch_tool** utility and list the nch objects:

```
nch_tool
nch_tool> listobjects *
```

The output from this command displays a list of all NCH object names on your system, similar to the following partial list:

```
nch_tool> listobjects *
Bes:bandit:FileNet
DefaultIMS:bandit:FileNet
IndexServer:bandit:FileNet
TapeServer1:bandit:FileNet
...
```

Use the listproperties command to inspect the properties associated with an object name:

nch_tool> **listproperties TapeServer1:bandit:FileNet**

Exit nch_tool when you are finished.

nch_tool> **exit**

If you plan to run your script on multiple systems each having a different disk server host, you can specify the disk server host as a parameter in your script. Then when you run the script, specify the disk server host name on the command line. For example, if your script specifies the following line in the DATASETS section:

location = $disk_location;

the disk server host name you specify on the command line when you start the script is substituted for the **$disk_location** parameter.

Note Command line parameters specified within a script always start with a dollar sign ($). However, omit the dollar sign when you specify a parameter name on the command line.

DEVICE_SPECIFICATIONS Section

In the DEVICE_SPECIFICATIONS section, you can define the backup or restore device to be a disk file, tape in a standalone tape drive, or tape in a tape library. The quickbu.tap script specifies the backup and restore devices as standalone tape drives using 8mm tape. The script assumes an AIX/6000 system with a rewinding tape device name of /dev/rmt0 and a non-rewinding tape device name of /dev/rmt0.1. Both must be specified. If the rewinding and non-rewinding device names on your system are different, use a text editor to change the names.

The location name is the host name or IP address of the host that owns the tape drive.

A backup or restore device as defined in this section is then identified for each thread in the THREADS section.

BACKUP_OPTIONS Section

The BACKUP_OPTIONS section specifies the type of backup (full or interval) and whether the backup will be online (FileNet software and databases are active) or offline (FileNet software and databases are shut down).

The backup options of the quickbu.tap script specify a full, online backup of the security database. Running an online backup does not disrupt operation of your system so you can try out the backup program even if production work is running.

THREADS Section

The THREADS section specifies the number of concurrent EBR threads and provides information about each thread. A one-to-one relationship exists between an EBR thread and a tape cartridge.

The quickbu.tap script uses one tape cartridge, hence only one EBR thread is specified. The device type specified for this thread is 8mm tape.

Running the Example Backup Script

You can run the EBR job in an X window (for UNIX environments) or DOS window (for Windows Server environments). The window should be at least 80 columns wide and 25 lines tall. (For running this example, you can shorten the window. If you are running more than two threads, increase the number of lines in the window to minimize scrolling of information out of the window area.) Precede the name of the script with an @ sign and follow the name with any parameter substitutions you have.

To run EBR in a non-X window environment, you must redirect the output to a tty device, such as the system console, or a file. When you run EBR as a **background** job, you must also redirect the output to a tty device. Do not redirect the output to a file if you run EBR as a background job. Otherwise the background job will stop with SIGTTOUT. However, you do not need to run EBR as a background job if you schedule the backup as a crontab job. A crontab job cannot be canceled because the job is issued from a cron daemon process. To terminate a crontab job, use the UNIX kill command.

If you do not specify any parameters, EBR displays extensive help text on stdout (the standard output device).

If you edit the example script, use the **–syntax** option of the EBR command to check your script for syntax errors before you start the backup. (See **Chapter 5, "Developing Your Scripts," on page 147** for more information.) For example, the command to check syntax of the Quick Start backup script is:

EBR –syntax @quickbu.tap

Enter the following command at the shell prompt to run the backup script:

EBR @quickbu.tap

Using Command Line Parameters

If you use command line parameters, commands may be long. UNIX users who cannot fit the entire command on a single line must use a backslash (\) continuation character as shown below:

```
EBR @cache.bac \
    location="TapeServer1:bandit:FileNet" ' \
    vg=TRAN_cache ser=T001 type=8mm \
```

This command assumes you are using the tape drive locally connected to the system containing the security database and the three-part NCH name of that tape drive is TapeServer1:bandit:FileNet. The command line parameters you use should reflect your system names.

In a process called substitution, command line parameter values are inserted into the script as if they had been written there. For example, for the disk service name, the effect would be the same as if the script line had been written as follows:

```
location = "TapeServer1:bandit:FileNet";
```

If you start EBR with an empty tape drive, the first two lines of the user interface display instruct you to insert a tape on a particular tape drive of a particular system. The instructions are in capital letters to make the message stand out. After you insert the tape, the backup program recognizes the tape and finishes the backup automatically—you do not have to respond to any prompts.

Note Double quotes must be part of the value of command parameters. However, the UNIX shell automatically strips off double quotes it finds in a command. To prevent the shell from stripping off the double quotes, enclose command line parameters in single apostrophes. The UNIX shell strips off the apostrophes, leaving the required double quotes in place.

Windows Server users entering commands in a DOS shell must use a backslash character (\) before each double quote surrounding the parameter as shown in the following example:

EBR @quickbu.tap $disk_location=\“TapeServer1:iqant6:FileNet\”

Viewing the Output

When the backup completes, EBR displays the progress log name. At the completion of each backup, check the progress log, the summary log, and the system event log. (See **“Status Reporting” on page 63** for details and directory locations of progress, summary, and system event logs.)

You may also want to run the EBR_tdir program to view the tape volume label:

EBR_tdir tape=/dev/rmt0 type=8mm

For more information, see **"EBR_tdir" on page 345**.

Tip: You can cause the tape to eject (or not) when executing the EBR_tdir command by using the eject=yes|no option. The default option is **yes**.

Re-running the Script

If you run the backup program again at this time, or if your first backup attempt failed and you try to re-run it, the backup will terminate with an error because your backup tape has not yet expired. (The expiration specified in the quickbu.tap script is one day.) However, you can reuse the tape immediately if you run EBR_label specifying the –erasedata option:

EBR_label tape=/dev/rmt0 type=8mm –erasedata

This option preserves the tape serial number, volume group name, and backup count. You can verify these preserved values by running EBR_tdir.

Quick Start Using a Disk File

Before running Quick Start using a magnetic disk file, you must perform some tasks to set up your environment. (See **"Preparation" on page 74**.) Since almost all functionality of Quick Start using tape as the

backup device applies to Quick Start using a magnetic disk file, only the differences are presented. Use the quickbu.dsk script below:

```
EBR_script( format_level = 2; );

BACKUP_GLOBAL_PARAMETERS
     volume_group = VG1;           -- tape volume group name
     expiration = 1 day;
END_BACKUP_GLOBAL_PARAMETERS

(continued on next page)
```

```
(continued from previous page)

DATASETS
     sec: MKF
        location = "demo";         -- hostname of security database
        base_data_file = "/fnsw/dev/1/sec_db0";
     end_MKF
END_DATASETS

DEVICE_SPECIFICATIONS
  DISK_FILE1 : disk_file
    location = "demo";
    filename = "/fnsw/local/tmp/diskbu1";
  end_disk_file
END_DEVICE_SPECIFICATIONS

BACKUP_OPTIONS
     sec: backup_options
        full_backup;
        online_backup;
     end_backup_options
END_BACKUP_OPTIONS

THREADS
     num_threads = 1;
     thread 1
        device = DISK_FILE1;
        volume_serial = R1;
        datasets
            sec;
     end_thread
END_THREADS
```

Make a copy of the script, placing it in a directory of your choice. Edit your copy of the script, specifying the full path name for the backup file in the THREADS section backup_device statement. (In the

quickbu.dsk script for a UNIX platform, the full path name for the backup file is /tmp/fake_tape.)

Use EBR_label to create and label the disk file. For example:

EBR_label disk=/tmp/fake_tape ser=R1 vg=VG1

Assuming you chose “bu_full.dsk” as the name of your copy of the script, start the backup with the following command:

EBR @bu_full.dsk

If you use command line parameters, parameter substitution and running the backup are the same as for Quick Start using tape. The name of the script will be different.

When the backup completes, view the progress log, summary log, and system event log, just as for the tape example.

To view the disk file label after the backup, run EBR_tdir against the backup disk file specifying the full path name for the backup file. For example, enter the following command to examine the disk file label for the Quick Start disk backup file:

EBR_tdir disk=/tmp/fake_tape

4

Developing Your Backup Strategy

A good backup strategy is essential to protecting your data. This chapter provides information to help you develop your backup strategy and covers the following topics:

- What to back up
- Recommended backup schedule
- Full backup versus interval backup
- Online versus offline backup
- Centralized versus decentralized backup
- Single versus multiple systems
- Immediate versus scheduled backup
- Constraints associated with certain datasets
- Strategies for using tapes

Note EBR operates only with servers that run FileNet software. For example, you cannot use EBR to back up local magnetic disks on personal computers running DOS/Windows.

What to Back Up

Using EBR, you can back up and restore the following:

- MKF databases
- Oracle index database
- Some locked page cache objects
- Raw partitions

You must decide where to run the user interface display and establish a window in which to run EBR. (See **“Establishing User Interface Display Windows” on page 64**).

Important On Magnetic Storage and Retrieval (MSAR) systems, the MSAR surfaces **must** be backed up at the same time as the EBR backup to keep the databases and surface files synchronized. The MSAR libraries **must** be put into backup mode before the EBR backup starts.

Use your preferred third-party backup method to backup MSAR surfaces; EBR does not support backing up MSAR surfaces.

The following table lists dataset types in a FileNet system. The table briefly describes each dataset and indicates whether backup is

required or optional. Detailed descriptions of each dataset type follow the table.

Overview of FileNet Datasets and Cache

Dataset	Type	Description	Required	Optional
Security database	MKF	FileNet security information	[	
Permanent database	MKF	Address information for each document	[	
Transient database	MKF	Information on work in progress (such as batch status, read/write and print requests), images in cache, and available cache space		[(Required if you back up cache or system does not have storage media)
Cache	Raw partition (UNIX) or file (Windows Server)	Different types of cache objects, including uncommitted images and pages waiting to be printed		[Required if you back up the transient database or system does not have storage media

Overview of FileNet Datasets and Cache, Continued

Dataset	Type	Description	Required	Optional
Index database	Oracle	Indexing fields, document classes, document indexing status, folder information, and WorkFlo queues (If you have an application server that contains SQL or WorkFlo Queue Services, the SQL user data and WorkFlo queues reside in another Oracle database on the application server)	[Back up in conjunction with MKF permanent database backup	
Oracle redo logs	Oracle	Transaction entries for the Oracle index database	[For Oracle redo log backup information, refer to **"Appendix J – Restoring Oracle" on page 411**	

Databases

EBR backs up MKF databases and the FileNet databases managed by the Oracle RDBMS.

EBR automatically backs up all necessary components of a database when you specify the database name. When you use EBR to back up an MKF database, EBR automatically backs up part of the MKF recovery log if necessary. You do not need to specify the MKF recovery logs in your backup script. Likewise, when backing up an Oracle index database, EBR automatically backs up the control files and the Oracle redo logs if necessary.

The sections below describe backup of the following databases:

- MKF databases and their associated MKF recovery logs
- Oracle databases and their associated Oracle redo logs

MKF Databases

The backup recommendation for MKF databases is weekly full backups and one or two daily interval backups between full backups. You **must** perform a full backup before you can perform the first interval backup of a database. If the amount of database activity is large, consider performing several interval backups of your MKF databases each day.

Note For offline backup of an MKF database, you first shut down the database. All changes in the MKF recovery log for the database are flushed to the database by the shutdown operation so a backup of the MKF recovery log is unnecessary.

For online backup of an MKF database, the system continues to run during the backup. Only the portion of the MKF recovery log that is generated during the online backup is necessary for recovery so EBR **automatically** backs up that portion of the MKF recovery log during the online backup.

If an MKF database restore becomes necessary, restore the last full backup followed by the **most recent** interval backup. EBR automatically rolls the databases forward to the end of the last transaction. The rollforward will be for less than a day's worth of processing if you perform interval backups at least once a day. The interval restore and the rollforward should occur very quickly.

CAUTION If you have modified the MKF database configuration by adding or removing MKF partitions or MKF recovery logs, do **not** perform an **interval** backup of the MKF databases until you have performed a full backup first. Otherwise, you will not be able to restore the interval backup.

MKF databases used by FileNet software include the transient database, the permanent database, the security database, and the network clearinghouse (NCH) database. Descriptions of each follow.

Transient Database

The transient database must always be backed up or restored in conjunction with a backup or restore of its associated cache. (See **"Cache" on page 101**.) EBR supports only full offline backups and full restores of the transient database.

Permanent Database

The permanent database, which maps document numbers to storage media locations, has one row per document and is generally the largest of the MKF databases. If your permanent database is large and you have multiple tape drives, consider striping the Perm_DB dataset. The other databases are generally much smaller and do not need to be striped.

Security Database

The security database contains FileNet security information and is generally small.

NCH Database

The NCH database maps software services symbolic names to network addresses so the system can perform remote procedure calls (RPCs). **The NCH database must be correct and uncorrupted for EBR to operate.**

CAUTION If the NCH database becomes corrupted, you cannot restore it with EBR. Do **not** use EBR to back up the NCH database.

You can easily and quickly recreate the NCH database by means other than backup and restore. The procedure below is one method for doing so.

Recreating the NCH Database

Use the following procedure to recreate the NCH database.

1. On Windows Server systems, shut down Image Services:

   initfnsw -y stop

   It is not necessary to shut down Image Services on UNIX systems but it is advisable.

2. Log on as a member of the fnadmin group if you are not currently so.

   To perform this procedure, you **be** a member of the fnadmin group.

3. At the system prompt, enter the following command to initialize the NCH database:

   fn_util initnch

4 Now that the NCH database is initialized, you must give the database an identity. Enter the following at the system prompt:

nch_update <domain name:organization name>

5 Enter the following command to establish the licensing environment:

lic_admin –f <license_filename>

where <license_filename> is the name of the file containing your software product license information. If your license file is not in your current working directory, use the full path name of the license file.

6 Repopulate the NCH database. Enter the following at the system prompt:

nch_tool <full path to the nch_dbinit file>

The full path to the nch_dbinit file is typically /fnsw/local/sd/nch_dbinit on UNIX systems and <drive>:\fnsw_loc\sd\nch_dbinit on Windows Server.

7 Make sure your NCH database was rebuilt. List out the NCH by entering the following at the system prompt:

nch_tool
nch_tool> **listobj ***
nch_tool> **quit**

You should see a complete list of NCH database objects.

Rebuilding WorkFlo Queue Entries

The procedure titled **"Recreating the NCH Database" on page 95** does not restore any WorkFlo queue information that was in the NCH database. To rebuild NCH entries for WorkFlo queues, a service representative uses the following sequence of WQS_tool commands:

allowupdates

qfunc

qnch <wsname> <qname>

The allowupdates command requires a password that your service representative must provide. The <wsname> is the workspace name and <qname> is the WorkFlo queue name you want to add to the NCH database. Specifications for <wsname> and <qname> options of the qnch command can contain wildcard characters (*). For information on WQS_tool, see the online help or the *Image Services System Tools Reference Manual* or your service representative.

Oracle Databases

Important The Oracle databases used by the FileNet system are the index database and WorkFlo Queue databases. EBR supports backup and restore of only these Oracle databases. EBR does not back up or restore other Oracle databases on your system.

EBR automatically backs up all necessary components of a database when you specify the database name. When you use EBR to back up an Oracle index database, EBR automatically backs up the control files and portions of the Oracle redo logs as necessary.

Index Database

The index database stores document attributes for query purposes and contains one row for every document on the system. The index database can be one of the largest datasets on the system. If your index database is very large, and if you have more than one tape drive available, consider striping the index database and backing up the database to multiple tape drives in parallel.

WorkFlo Queue Database

WorkFlo Queue databases are generally very small and these small datasets should not be striped. If only one WorkFlo Queue database exists and is on the same host as the index database, then it is part of the same Oracle database. Otherwise, the WorkFlo Queue database is a separate Oracle database.

Online Backups Of Oracle Databases

If you plan to perform online backups of Oracle databases, you **must** enable archive log mode. In addition, make sure Oracle automatic archival is enabled. This feature prevents your database from hanging due to full online redo logs. Even if you plan to perform offline backups, we recommend that you turn on archive log mode. After a restore, Oracle databases cannot be rolled forward to the last transaction if archive log mode is not enabled. If the rollforward recovery does not take place, the restored Oracle database becomes unsynchronized with other FileNet datasets. Unsynchronized databases occur when databases are not updated to the same point in time. This condition causes significant problems that usually require technical support assistance to resolve.

CAUTION Do not randomly turn archive log mode off and on. Doing so may prevent a full recovery of Oracle databases during a restore. Select the archive log mode necessary for your environment and leave the mode at that setting.

The directory into which archive redo logs are written is specified in the log_archive_dest Oracle initialization parameter. Refer to **"Appendix F – Enabling Archive Log Mode" on page 377** and your *Oracle7 Server Administrator's Guide* or *Oracle8 Server Backup and Recovery Guide* for more information.

In addition to weekly full backups, one or more daily interval backups are recommended for Oracle databases. You **must**, however, perform a full backup before you can perform the first interval backup of the database.

Oracle Signature File

EBR requires the presence of a signature file to support interval backups of Oracle databases. The signature file stores a page signature for every page in the Oracle database. When a page in the database changes, the page signature changes. EBR uses the page signature to determine which pages in the database have changed since the last full backup or last restore. Only pages with a changed signature are backed up or restored.

The signature file (oraIDB_sig) resides in a signature file directory which you create before backing up Oracle. You specify the directory name in your backup script (examples in this document specify the directory as /fnsw/local/tmp/ora_sig/). Every successful full backup and full restore creates a new copy of the signature file then deletes the old copy. Before the deletion of the old copy takes place, both the old and new signature files exist in the signature directory. When allocating

space to the directory, allow 3.2% of the Oracle database size to hold the old and new copies of the signature file. We recommend that you do not store other files in the signature file directory.

Recovery and Redo Logs

MKF recovery logs and Oracle redo logs perform the same function for their respective databases—record database transactions that are used to roll a database forward to the last transaction if a restore becomes necessary.

When you use EBR to back up an MKF database, EBR automatically backs up part of the MKF recovery log if necessary. You do not need to specify the MKF recovery logs in your backup script. Likewise, when you use EBR to back up an Oracle index database, EBR automatically backs up the control files and some of the Oracle redo logs as necessary.

When planning your backup strategy, be aware of the importance of recovery and redo log magnetic media space allocation.

For MKF database recovery logs, allocate enough magnetic media space to hold a minimum of two days' worth of processing. If archive log mode is enabled for the Oracle index database, allocate sufficient space to hold the Oracle archive redo logs.

Use the following guidelines to determine how much log information your system produces:

- For MKF databases, perform the procedure described in **"Appendix E – Computing MKF Log Space" on page 374**.
- For Oracle databases, monitor the total size of archived redo logs produced in 24 hours. Allocate enough magnetic media space to

hold at least the amount of data that is generated between your backups.

For more information, see **"Database Archive Log Issues" on page 109**.

Cache

The FileNet system requires that certain FileNet datasets be synchronized, which means that all datasets must be updated to exactly the same point in time. The MKF transient database and its associated cache are a tightly-coupled pair of datasets—they must always be perfectly synchronized. The magnetic disk cache contains data objects (for example, compressed images that have not been written to storage media) and available space. The transient database contains mapping and space use information for the objects in cache. (See **"Transient Database" on page 94**.) Always back up and restore cache and the associated transient database together.

EBR supports only full offline backups for cache and the transient database. Therefore, EBR does not support interval restores of cache or the transient database.

Note EBR does not support backing up or restoring cache objects on a remote BES cache server.

As described in **"Recovery and Redo Logs" on page 100**, a database recovery log gives a database its rollforward capability. Magnetic disk cache has no recovery log so, unlike a database, cache cannot be rolled forward. Since cache cannot be rolled forward, EBR must not, and does not, roll the transient database forward either.

Your strategy for backing up and restoring cache and the transient database is affected by your system implementation of a storage library and optical storage media. The following sections identify strategy considerations for systems with storage libraries and optical storage media and systems without storage libraries and optical storage media (cache-only systems).

Systems with Optical/MSAR Storage Media

The magnetic disk cache contains work in progress. Even if the transient database and the locked objects in cache are backed up, they cannot be rolled forward after a restore. All processing since the last backup is lost if you have to restore.

Disk mirroring is the most effective way to protect the scanned batches prior to their committal to optical storage media. We recommend that you mirror all your magnetic disks, as explained in **Chapter 1, "Introduction," on page 26**, but mirroring is especially important for cache because of the lack of rollforward recovery capability.

You could back up the cache daily along with the transient database. However, due to the large size of the cache, doing so is generally inconvenient. Furthermore, the system must be shut down during the backup of cache and the transient database.

Disk mirroring becomes even more important in a 7-by-24 environment (a system that runs seven days a week and twenty-four hours a day). In this environment, no opportunities exist for a system shutdown for backup and restore. You may choose not to back up the cache and transient database, relying instead on disk mirroring for protection.

Systems without Optical Storage Media

For systems without optical storage media, cache contains not only work in progress but all the scanned document images. Therefore, for a cache-only system, you **must** perform a daily backup of cache and the transient database. Since the system must be shut down during cache and transient database backups, 7-by-24 operation is not possible for a cache-only system.

For a cache-only system, the recommended cache backup strategy is:

- Perform a full offline backup once a week of both the transient database and the locked objects in cache.
- Perform a full offline backup once a day of both the transient database and the locked objects in cache.

This backup strategy also applies if your system has optical storage media but certain documents are never migrated from cache to storage media.

For information on converting a cache-only system to an MSAR (Magetic Storage and Retrieval) system, see *MSAR Procedures and Guidelines*. To download IBM FileNet documentation from the IBM support page, see **Accessing IBM FileNet Documentation**.

Raw Disk Partitions

The FileNet system stores database data files, database recovery logs, and cache data in raw disk partitions. You may also have non-FileNet data in raw partitions that you want to specifically back up as raw partitions.

Note When backing up FileNet datasets, you must specify the correct dataset type (MKF, Oracle, or cache) in your script.

On UNIX platforms, you can use EBR to back up raw partitions. (The Windows Server platform does not support raw partitions.) In your backup script, simply specify the dataset type as **partition**. You must also make sure that the raw partition is not in use before you start a backup or restore of the partition.

CAUTION Do not back up a directory by defining it as a partition. EBR will back up a directory if you define it in your script as a raw partition. However, when you attempt to restore that raw partition, EBR restores it as a file, not a directory. EBR does not support backing up files and directories. See **"Other Files" on page 104** for more information.

Other Files

EBR supports backup of cache, raw partitions, MKF databases, and the Oracle index database and its online redo logs. EBR cannot back up subdirectory trees or other files, for example, FileNet file system (/fnsw) files, including configuration database files, Oracle archive logs, and data files not in raw partitions. (Oracle archive logs are stored in the directory you specified in the log_archive_dest Oracle initialization parameter.) For files not supported by EBR, use another backup method, such as the tar (tape archive) utility, to copy the files to magnetic tape or magnetic disk.

Recommended Backup Schedule

We recommend a backup schedule that includes full backups once a week and interval backups once or twice a day. The best time to perform the full backup is when your system is least busy.

Note You **must** perform a full backup before you can perform the first interval backup. See **"Full versus Interval Backups" on page 107**.

Performing interval backups twice a day reduces magnetic media space requirements for database recovery logs (both MKF recovery logs and Oracle redo logs). Twice-a-day interval backups also reduce restore time, since the amount of database rollforward log information that needs to be processed for the final rollforward phase is also reduced.

We recommend that, at a minimum, you follow this backup schedule for your datasets:

- Weekly full backups of the security database and permanent database.
- Weekly full backups of Oracle databases (the index database and WorkFlo queue databases).
- Daily interval backups of the security database and permanent database. If the amount of database activity is large, consider performing several interval backups of your MKF databases each day.
- One or two daily interval backups of Oracle databases (the index database and WorkFlo queue databases). If the amount of index database activity is large, consider performing several interval backups each day.

The primary protection mechanism for the transient database and cache is disk mirroring.

Full versus Interval Backups

You can perform full or interval backups of the MKF permanent and security databases and the Oracle index database. Cache and transient database backups must always be full, offline backups.

Before you can perform the first interval backup, you must perform a full backup of the database. After the full backup, each subsequent interval backup captures only the data that has changed since the last full backup. Interval backups are cumulative so a restore always consists of restoring the last full backup followed by the most recent interval backup.

Online versus Offline Backups

You can perform full and interval backups either online or offline with the following exception: backups of cache and the transient database must be full offline backups.

Installations that operate in 7-by-24 mode must perform online backups because the system is never shut down.

When you shut down the FileNet system prior to an offline backup, all changes recorded in the MKF recovery logs are flushed to the database. Since no data remains in the recovery log at that point, EBR does not back up the MKF recovery logs.

For an online backup, EBR backs up only the portion of the MKF recovery log that was generated **during** the online backup. EBR automatically uses this portion of the log during restore processing to return the MKF database to a usable state.

Database Archive Log Issues

This section describes archive transaction log issues of which you should be aware when performing database backup and restore operations. Issues vary depending on the database management system (MKF or Oracle).

MKF Archive Recovery Log Issues

The MKF recovery log is never archived—it is simply overwritten. If the MKF recovery log is overwritten before a backup is performed, a detailed message is written to the system event log. The message explains that the MKF database cannot be rolled forward from the last backup because insufficient magnetic media space was allocated to the MKF recovery log.

If you run daily backups as recommended, magnetic media space requirements for MKF recovery logs (and Oracle redo logs) will be approximately constant.

If you perform daily backups, recovery and redo log information recorded prior to the last backup is redundant. Normally, you only need the log information produced after the last interval backup. Only in the event that you damage or lose the backup tapes or skip a backup do you need older recovery log and redo log information.

Oracle Archive Redo Log Issues

Oracle is capable of automatically making a copy of its online redo log and writing it to the directory you specify at installation time (for example, /fnsw/local/tmp/archivelogs/ARCH). This copy is called an archive redo log. You must activate Oracle archive log mode to have

Oracle perform this automatic archiving of its online redo logs to magnetic media.

If you do online backups and you use Oracle, you **must** archive the redo logs to magnetic media. Even if you normally perform offline backups, you should turn on Oracle's archive log mode. The archive logs are required after a restore to roll the database forward to the last transaction. If archive log mode is not enabled, the lack of archive log transaction entries prevent the index database from being rolled forward after a restore. Database updates are lost and the index database is unsynchronized with other datasets on the system.

To turn on archive log mode, see **"Appendix F – Enabling Archive Log Mode" on page 377** or see your Oracle documentation.

CAUTION Do not randomly turn archive log mode off and on. Doing so may prevent a full recovery of Oracle databases during a restore. Select the archive log mode necessary for your environment and leave the mode at that setting.

If you run daily backups, you do not have to worry about your magnetic media filling up with recovery logs for MKF or redo logs for Oracle databases. You do not need to copy MKF recovery logs or Oracle redo logs to tape. EBR manages these logs for you as described below.

Whenever you perform a backup of the Oracle database, EBR automatically deletes Oracle archive logs older than the number of days you specify in the archive_redo_log_retention parameter of your backup scripts. This retention period is the number of days EBR retains Oracle archive logs before automatically deleting them. If you find that the archive logs are consuming too much magnetic media space, reduce the number of days EBR retains them. For example, if

you specify 14 days as the retention period, and two weeks' worth of Oracle archive logs consumes too much space, decrease the number of days in the archive_redo_log_retention parameter of your backup scripts. The minimum number of days you can specify is two.

CAUTION The **minimum** number of days of retention for archive logs is equal to the number of days since the last full backup.

Centralized versus Decentralized Backups

Decide whether to take a centralized, decentralized, or mixed approach to backup. Remove backup tapes from the vicinity of the computer as soon as possible. If a site disaster occurs, the backup tapes are unaffected. The probability of a site disaster might affect your choice of centralized versus decentralized backups.

You could take a decentralized approach by backing up each system to its own local tape drives. With a decentralized approach, tape cartridges must be inserted on each system and the data is not transmitted over a network to a central system.

You could take a centralized approach by backing up multiple systems over the network to a central backup system that has multiple tape drives. With a centralized approach, tape cartridges only need to be inserted on the central system and data is transmitted over a network.

You can also use a combination of these approaches.

Single versus Multiple Systems Per Tape

You can back up several systems to a single set of tapes or you can back up each system to its own set of tapes. We recommend backing up each system to its own set of tapes for the following reasons:

- The data for each system is separate, not intermingled.
- Restores are faster.
- Backing up multiple systems to a single set of tapes tends to make the backup tapes too valuable by putting too much data on them. Damage to or loss of this single tape set jeopardizes the status of multiple systems.

Immediate versus Delayed Backups

An immediate backup begins as soon as you submit your EBR backup script. You can also schedule a delayed backup (also called a scheduled or deferred backup), which occurs at some future time you specify in your script. Delayed backups have certain benefits, as described in the topics below.

Benefits of the Delayed Backup Feature

Delayed backups occur at a specific time in the future. The delayed backup feature coupled with the use of other script parameters enables all the backups for a week or even a month to be initiated at once with a shell script running in an X window.

You can implement delayed backups using the start_date and start_time global parameters in your EBR backup script.

Tip In a UNIX environment, using the crontab utility is the preferred method of running delayed backups. If you are in a Windows Server environment, crontab is not available. Use start_date and start_time or the Windows Server AT command to schedule delayed backups. (See **"UNIX crontab Utility" on page 116** and **"Windows Server AT Command" on page 118** for more information.)

Delayed Backup Example

Suppose you created a full backup script file named "fullback" which uses only one tape drive and has four parameters:

- $sday for the global start_date parameter
- $stime for the global start_time parameter

- $ser for the volume_serial parameter
- $vg for the global volume_group parameter

You also created an interval backup script file named “ivalback” which uses one tape drive and has four parameters:

- $sday for the global start_date parameter
- $stime for the global start_time parameter
- $ser for the tape_serial parameter
- $vg for the global volume_group parameter

The following shell script sequentially initiates a week’s worth of backups in a single shell window:

```
EBR @fullback sday=2003/12/22 stime=16:30 ser=BDT961209001 vg=BANDIT_
SUNPM3_ONLFULL; EBR @ivalback sday=2003/12/23 stime=16:30
ser=BDT961209002 vg=BANDIT_MONPM3_ONLIVAL; EBR @ivalback sday=2003/12/24
stime=16:30 ser=BDT961209003 vg=BANDIT_TUEPM3_ONLIVAL; EBR @ivalback
sday=2003/12/25 stime=16:30 ser=BDT961209004 vg=BANDIT_WEDPM3_ONLIVAL;
EBR @ivalback sday=2003/12/26 stime=16:30 ser=BDT961209005 vg=BANDIT_
THUPM3_ONLIVAL; EBR @ivalback sday=2003/12/27 stime=16:30
ser=BDT961209006 vg=BANDIT_FRIPM3_ONLIVAL; EBR @ivalback sday=2003/12/28
stime=16:30 ser=BDT961209007 vg=BANDIT_SATPM3_ONLIVAL;
```

The @ character preceding the “fullback” and “ivalback” names in the example above indicates that these are script file names.

All that remains is to put the cartridge in the tape drive before 4:30 p.m. each day and check the results of the backup. All the backups start sequentially at the appropriate date and time and display current status in the X window.

Note If you recycle Image Services software or reboot your system before the scheduled jobs run, the EBR delayed backup jobs are lost and must be rescheduled. (A software recycle or system reboot does not affect jobs scheduled by the UNIX cron command or Windows Server AT command.)

In addition, any user with proper permissions can delete all jobs from the scheduler.

For details about the global parameters used to create scheduled backups, see **"BACKUP_GLOBAL_PARAMETERS Section" on page 182**.

Alternate Delayed Backup Methods

The crontab utility is the preferred method of running delayed backups in a UNIX environment. In a Windows Server environment, use the AT command to schedule delayed backups. Each method is briefly described below.

UNIX crontab Utility

UNIX

The crontab utility is the preferred method for performing delayed, unattended EBR backups in a UNIX environment. Because delayed job information is kept on magnetic media, crontab jobs start up at the appropriate time even after a recycle of Image Services software or a system reboot. In contrast, jobs listed in a shell script using EBR start_date and start_time could be lost after a recycle of Image Services software or a system reboot. For example, if you must perform an offline backup before the start_date and start_time for the delayed backup, the delayed EBR backup jobs are terminated when the server is shut down.

When implementing delayed backups, be aware of the following operational requirements:

- The EBR user interface window must remain open until the delayed backup operation completes.
- If you shut down or recyle your FileNet system before the delayed backup jobs execute, the delayed backup jobs are terminated.

To use the crontab utility, you must either be logged in as the root user or your login ID must be included in the /var/adm/cron/cron.allow file of your UNIX system. (The directory name may be different on your system.) The cron.allow file contains an entry for each login ID that has permission to run crontab. If your login ID is not in cron.allow, the following message displays when you attempt to run crontab:

> You are not authorized to use the cron command.

Use the **crontab –e** command to create and edit the crontab file with the conditions under which you want the backup job to start. Each crontab file entry consists of a line with the following six fields:

- The minute (0 through 59)
- The hour (0 through 23)
- The day of the month (1 through 31)
- The month of the year (1 through 12)
- The day of the week (0 through 6 for Sunday through Saturday)
- The shell command to start EBR

At the specified date and time, the cron daemon runs the shell command named in the sixth field. The shell command must include an

option to open an X window (for example, xterm –e). The following command set from the FileNet-supplied cron.sh shell script is one example:

```
X_TERM=xterm
SYSTEM=demo
DISPLAY=${SYSTEM} : 0
$ {X_TERM} -d ${DISPLAY} -e /fnsw/local/EBR/samples/run_backup.sh
```

If you are using one of the FileNet-supplied sample scripts, such as cron.sh, or a script that follows the example of the supplied scripts, an X window opens when EBR starts and the progress log displays in this window. When the job completes, press the Enter key to close the window.

For more information on creating crontab files, running the crontab command, or scheduling jobs with the crontab utility, refer to your operating system's online manual (man) pages.

Windows Server AT Command

You can use the Windows Server AT command as an alternative method of scheduling delayed backups. To schedule jobs, you must enable the Windows Server Schedule service from the Windows Server control panel. If you attempt to use the AT command before the service is enabled, Windows Server issues a message similar to the following:

> The service has not been started.

For more information on using the Windows Server AT command, refer to the Windows Server online help or see the **run_bkup.cmd** file in the

\fnsw_loc\\EBR\samples directory. The run_bkup.cmd file contains AT command examples.

The following AT command examples schedule backups for the system named iqant6. The first command schedules daily online interval backups to occur at 4:30 p.m. Monday through Friday and the second command schedules a weekly full online backup to occur at 4:30 p.m. every Sunday:

AT \\iqant6 16:30 /interactive /EVERY:Monday,Tuesday, Wednesday,Thursday,Friday D:\fnsw_loc\EBR\samples\ival_on.cmd

AT \\iqant6 16:30 /interactive /EVERY:Sunday D:\fnsw_loc\EBR\samples\full_on.cmd

The /interactive parameter opens a window in which the EBR progress log displays when the job starts. FileNet-supplied sample scripts or scripts that follow the example of the supplied scripts use the /interactive parameter. When the scheduled job completes, press any key to close the window.

See your operating system online help for details of the AT command.

Unattended Backups

Most delayed backups are run in **unattended** mode. In an unattended backup, the backup tape is preloaded in the tape drive and no operator is present during the backup.

To schedule an unattended backup, use the **crontab utility** in a UNIX environment or the **AT command** in a Windows Server environment.

Important

On Magnetic Storage and Retrieval (MSAR) systems, the MSAR libraries **must** be put into backup mode before the EBR backup starts. To do this, you can create a script to start DOC_tool and set the mode. The MSAR surfaces **must** be backed up at the same time as the EBR backup to keep the databases and surface files synchronized.

Use your preferred third-party backup method to backup MSAR surfaces; EBR does not support backing up MSAR surfaces.

UNIX Unattended Backups

UNIX

To run EBR in an X window environment and invoke EBR scripts, use the xterm –e command. The –e option starts an xterm window and executes EBR and the EBR script file. The following example starts an X window and executes the EBR script, daily.bac :

```
Display=demo:0
Xterm –d ${DISPLAY} –e /fnsw/bin/EBR@/usr/opr/daily.bac
```

To run EBR in a non-X window environment, you must redirect the output to a tty device, such as the system console, or a file. When you run EBR as a **background** job, you must also redirect the output to a tty device. Do not redirect the output to a file if you run EBR as a back-

ground job. Otherwise the background job will stop with SIGTTOUT. However, you do not need to run EBR as a background job if you schedule the backup as a crontab job.

The following is an example of redirecting output to a tty device:

```
/fnsw/bin/EBR@/usr/opr/daily.bac>/dev/console
```

The following is an example of redirecting output to a file:

```
/fnsw/bin/EBR@/usr/opr/daily.bac>/fnsw/local/logs/backup.log
```

To run the EBR job in console mode, you can add the commands such as in the above examples in a shell script file, for example, /usr/opr/backup.sh, to be used by the cron daemon.

You cannot cancel a crontab job because the job is issued from a cron daemon process. To terminate a crontab job, use the UNIX kill command.

Example Crontab Job

To schedule an unattended backup every day at 23:00 pm, use the crontab –e command to add a cron job entry. Then enter the cron job entry. Each entry contains six fields separated by spaces or tabs, as shown in the following format:

```
minute hour day_of_month month weekday command
```

An asterisk means all values are allowed. For example, to run a job every day, specify asterisks in the day_of_month, month, and weekday fields. The following is an example of running backups every day at 23:00, using a shell script file (/usr/opr/backup.sh). Enter:

crontab –e

When the editor opens, enter:

0 23 * * * /usr/opr/backup.sh

When using the crontab utility to schedule unattended backup jobs, you must specify the full path name of all programs, script files, and include files within the EBR scripts. The full path name enables the cron daemon process to locate the files.

For additional information, view the sample backup shell script file located in /fnsw/lib/ebr/cron.sh.

Note You can use the UNIX **at** command to submit one-time backup jobs. Refer to your operating system online help for details.

Windows Server Unattended Backups

WIN

You can use the Windows Server **AT** command to schedule an unattended and/or delayed backup job. The AT command schedules commands and programs to run at a specified time and date. The AT command automatically starts EBR in a DOS window. See **"Windows Server AT Command" on page 118** for more information. Also refer to the Windows Server online help for details of the AT command.

For additional information, view the sample backup script file located in \fnsw\lib\ebr\run_bkup.cmd.

Constraints

When planning your backup and restore strategy and when developing your backup and restore scripts, be aware of the constraints that apply when using EBR. This section describes constraints associated with:

- The enforced order, called “order constraints,” in which EBR backs up and restores datasets when a cache is involved
- Backing up and restoring a magnetic disk cache and its corresponding transient database
- Too much backup data to fit on a single tape cartridge (referred to as “data overflow”)
- Appending data to backup tapes
- Memory requirements when using multiple threads

Order Constraints

When you back up or restore a cache, EBR enforces that certain dependent datasets are backed up and restored in a certain order. In this way, EBR protects data and maintains dataset synchronization. For example, the transient database and cache must always be synchronized, that is, updated to the same point in time. To assure that these two datasets remain synchronized, EBR requires the permanent, security, and transient databases to be backed up or restored before cache. The order in which EBR backs up or restores the databases themselves is random, but cache must always be the **last** dataset backed up or restored when you backup or restore the transient, permanent, or security databases as part of the same script.

The dataset order is maintained through the **order_constraints** statement in the DATASETS section of your backup or restore script. Specify this statement only when you are backing up or restoring a cache.

EBR enforces the following order constraints:

- Permanent database is backed up before cache
- Security database is backed up before cache
- Transient database is backed up before cache
- Cache must always be the **last** item in a list of datasets that includes cache and one or more of the following datasets: permanent database, security database, transient database

For the syntax of order_constraints statements, see **"Maintaining Cache Data Integrity through Order Constraints" on page 194**.

Backup Constraints for Cache and the Transient Database

A physical magnetic disk cache may contain more than one logical cache. During a cache backup, EBR backs up locked objects in logical caches **except** for any object with a document ID of greater than 3,999,999,999. Document IDs above this value are considered to be temporary objects created by:

- Fast batch committal
- The stdoccpy tool and other background jobs
- Print services

Print cache objects are viewed as temporary objects and never backed up. By the time a restore occurs, the print jobs will probably have finished. If not, you can resubmit the print requests.

EBR backs up only locked cache objects, ignoring objects that are not locked. After a restore, objects that were not locked can be retrieved from optical storage media.

Since eligible locked objects in cache are backed up regardless of the logical cache they are in, you must specify the name of only one of the logical caches. You can specify the three-part NCH name of the logical cache, such as page_cache1:costa10:FileNet, or simply the host name of the logical cache, such as costa10. In fact, if you specify more than one logical cache controlled by the same transient database, EBR generates a syntax error when you submit your script.

The following restrictions apply when you perform a backup of cache:

- You must perform a full backup of the associated transient database **in the same script.**
- Only full offline backups are supported for cache and the transient database.

Cache must be offline (not in use) while it is being backed up. Before you attempt to back up cache, use Task Manager Stop button to stop the FileNet software on the host containing the cache and then the Task Manager Backup button to put the system in **backup mode**. When the backup completes, use Task Manager Restart to restart the FileNet software. Refer to your *Image Services System Administrator's Handbook* for information about how to use Task Manager. To download IBM FileNet documentation from the IBM support page, see **Accessing IBM FileNet Documentation**.

Restore Constraints for Cache and the Transient Database

This section describes the requirement for simultaneously restoring the magnetic disk cache and the transient database, as well as some special handling of the transient database during a full restore of cache.

Simultaneous Restore of Cache and the Transient Database

When you perform a restore of a magnetic disk cache, you must perform a restore of the associated transient database **in the same script**.

As with all restore operations, use Task Manager to put the FileNet software into **restore mode** before starting your cache and transient database restore operation. From the Task Manager application, select the Stop button followed by the Restore button. When the restore completes, select the Task Manager Restart button to restart the FileNet software.

If a restore of the permanent database, security database, or transient databases is necessary at the same time as a cache restore, EBR enforces that the database restore operations complete before the cache restore starts. Cache must always be the **last** dataset to be restored. (See **"Order Constraints" on page 123**.)

Note If the rollforward of the MKF permanent database fails, usually due to corrupted recovery logs, you **must** run the SNT_update program before you attempt to restore the transient database and cache. For more information, see **"Scalar Numbers Table Update After Restore Failure" on page 259**.

Special Transient Database Handling

When you restore a full backup of cache, EBR enforces that the transient database is restored first. Special handling of the transient database is done on **full** restores of a cache, as described below.

Note Cache does not have a recovery log and cannot be rolled forward. You must restore the transient database and cache together to maintain synchronization between the datasets. If you restore the cache without also restoring the transient database, work is lost and the cache will no longer be synchronized with data on optical media.

Since cache has no rollforward capability, EBR ensures that restored cache objects remain synchronized with the transient database. EBR prevents the transient database from rolling forward, even if the recovery log splices in, by writing a zero block at a strategic location in the transient database recovery log.

When the cache is restored, EBR does the following:

- Restores cache objects to the magnetic disk referenced by the used space table of the transient database.
- Deletes background jobs, temporary objects and their associated write requests, unlocked objects, and print requests from the transient database.
- Excludes objects from the restore that have already been committed to optical media.

Using this strategy, EBR eliminates temporary objects that have presumably been processed and preserves batch information and write requests. EBR consolidates all steps of the cache recovery procedure and performs them automatically.

EBR deletes background jobs records from the MKF database tables after restoring cache and the transient database. However, EBR does not delete background job work files. Background job work files are in the following directories:

/fnsw/local/logs/bkglog	For UNIX platforms
<drive>:\fnsw_loc\logs\bkglog	For Windows Server platforms

Examples of background job work files are those generated by document copy jobs (CpyLog.xxx, CpySid.xxx, CpyUid.xxx), annotation copy jobs (AntLog.xxx), and import jobs (ImpLog.xxxxxx). To keep old background job work files from filling up magnetic media space, periodically delete these files.

Cache Restore Constraints for Multiple Server Systems

MultSv

If you are restoring to a system configured with multiple storage library servers or multiple application servers, one of the storage library servers is designated as the document locator server. The permanent database on the document locator server must be restored first so that cache restore on other servers in your configuration can access the necessary database tables for information about outstanding write requests to optical storage devices.

One approach is to restore all the MKF permanent databases before you restore cache. For example, if your configuration includes three storage library servers, the order_constraints statement in the DATASETS section of your restore script may resemble the following example:

```
order_constraints
    sec,perm1,perm2,perm3,tran1,tran2,tran3 before
```

```
    cache1,cache2,cache3;
end_order_constraints
```

Another approach to the same configuration might result in the following example:

```
order_constraints
    sec,perm1 before perm2,perm3;
    perm1,tran1,perm2,tran2,perm3,tran3 before
    cache1,cache2,cache3;
end_order_constraints
```

Data Overflow

EBR does not support data overflow, a condition in the data to back up will not fit onto a single tape cartridge. EBR does not automatically allow backup data to “overflow” onto additional tape cartridges. If your backup data exceeds the capacity of a single cartridge, use the stripping technique described in this section.

Tip EBR supports the Exabyte Model 8900 (“Mammoth”) tape drive which is a high-capacity tape drive. The use of higher capacity tape cartridges with the Model 8900 may also be a solution to a data overflow problem.

When you have more data to back up than fits on one cartridge, set up a backup script specifying that multiple threads share a drive. One advantage of this approach is that if you add additional tape drives, the backup tapes you already made can be restored concurrently from multiple tape drives. To implement this approach, take the following actions:

- Define more than one thread in the THREADS section of your backup and restore scripts.
- Assign the same tape drive to more than one thread in the THREADS section.
- Break large datasets into multiple parts (a procedure called “striping”) and assign each part to a different thread.

When the backup or restore runs, EBR accesses the threads in increasing thread number order.

If a single tape drive is shared by multiple threads, an operator must be available to switch tape cartridges. Since the cartridges are processed sequentially, the backup or restore takes much longer than if each part of a multi-part (striped) dataset is run concurrently using a different tape drive.

If a backup is made using multiple threads, that same set of backup tapes can later be restored using any number of compatible tape drives up to the number of threads used for backup. This protection can be invaluable if one or more tape drives becomes unavailable.

For example, if you normally back up using two drives and one drive fails, you can still back up to two cartridges by having the two threads share the remaining drive. Then, if you need to do a restore after the drive is repaired and made available, you can restore using two drives concurrently. If you need to do a restore before the drive is repaired, set up the threads in your restore script to share the available tape drive. Similarly, if you back up using two drives and one drive later breaks, you can still restore the backup by having the two restore threads share the working tape drive.

Appending to Backup Tapes

You cannot append an EBR backup to a previous EBR backup tape. Also, EBR does not support appending data to tapes created by third party backup products.

Memory Requirements When Using Multiple Threads

Running multiple threads requires more processor memory than running a single thread. Typically, you use between one and three threads. However, if your system has less than 32 MB of memory, do not attempt to use more than two threads.

To ensure that sufficient memory is available for the number of threads you want to use, refer to **"THREADS Section" on page 205** and **"Appendix G – Memory Requirements" on page 384**.

Developing a Strategy for Using Tapes

You must decide which tape drives to use on which systems. Consider the following information when selecting your tape devices and determining your tape configuration.

Selecting Tape Device Type

During a backup, EBR cannot distinguish between a tape that was prelabeled with a tape device type specification different from the device type you specify in your backup script. For example, you could prelabel a tape specifying a tape type of QIC and run a backup script (using that QIC prelabeled tape) that specifies a tape device type of 8mm. EBR does not compare the drive type specified in the script with the actual tape used for backup. Since many tape devices have some common features, EBR may successfully complete this backup. However, EBR may be unable to restore this tape.

CAUTION Be sure to specify the same tape device type in your backup script as you specified in the EBR_label command to prelabel the backup tape.

Interchanging Tapes

You can interchange tapes between systems of the same type (for example, between two AIX/6000 systems). Tapes generated by EBR cannot be interchanged between different platforms. For example, 4mm DAT tapes generated on an HP-UX system contain inter-record gaps that cannot be read by an AIX/6000 system or a Solaris system.

If you plan to interchange EBR tapes between like systems, you must set the tape drive block size on each of those systems to the same value.

In addition, the tape drives you use to create the tape and read the tape should be configured with the same characteristics (for example, density). Then most EBR tapes can be used on same-type systems.

Selecting Tape Densities

Tapes generated from a high density tape drive cannot be read on a low density tape drive. We recommend that you use the highest density available. To allow a low density tape to be read on a high density tape drive, specify a variable block size on both drives.

Using Quarter-Inch Cartridge Tape Drives

When insertion of a second tape into a tape drive becomes necessary, EBR issues internal commands to set the tape drive to offline status. QIC tape drives on Image Services servers do not recognize this command and do not automatically eject the current tape in the drive. Consequently, if you have a single shared QIC tape drive, and a second or subsequent tape must be inserted during a backup on an Image Services server, EBR issues messages indicating an incorrect tape is in the drive.

If this occurs, manually eject the tape from the tape drive and insert the next tape. EBR automatically recognizes the new tape on the next read attempt and proceeds with the operation.

If your platform is Windows Server and multiple threads share the same QIC tape drive, the EBR tape process attempts to read the same tape after the first tape completes. EBR issues an “incorrect tape inserted” message followed by a tape request message. These messages repeat until you manually eject the first tape and insert a new tape into the drive.

Note On a Windows Server system, if you manually eject a QIC tape from the drive and insert a new tape, EBR may terminate after generating a system error indicating no media exists in the tape drive. If this occurs, rerun your backup or restore operation.

Using the Exabyte Model 8900 (“Mammoth”) Tape Drive

The Exabyte Model 8900 (“Mammoth”) tape drive can write up to 20 GB of uncompressed data to each tape cartridge. (We recommend that you do not use compression provided by your tape drive hardware. See **“Data Compression” on page 58** for more information.) To obtain maximum performance with this high-capacity tape drive during EBR backups and restores, the tape drive must be locally attached to the host system from which you are initiating the backup or restore.

Specifying Tape Drive Device Names on Solaris Servers

SOL

In a Solaris environment, correct Berkeley Software Distribution (BSD) specification becomes important when you have non-EBR tape applications that require BSD or non-BSD tape device behavior. EBR allows you to specify **either** BSD or non-BSD tape device behavior and manages tape marks accordingly when reading tape files.

The format of a BSD device name in your EBR script is:

```
/dev/rmt/[0-127] [l,m,h,u,c] b [n]
```

The non-BSD device name format is:

```
/dev/rmt/[0-127] [l,m,h,u,c] [n]
```

Designate tape density with one of the following characters:

l = low
m = medium
h = high
u = ultra high
c = compressed

Specify BSD behavior with the **b** character in the tape device name. Omit the **b** from the device name if you need non-BSD device behavior.

The **n** character is optional and, if specified, indicates no-rewind-on-close behavior.

Note EBR checks both the rewind and no-rewind tape device names and selects the appropriate device. However, the EBR_label and EBR_tdir utilities require that you specify only one tape device name. You can specify either the rewind or no-rewind name.

Commonly-used tape device names in Solaris are actually **links** to BSD tape devices. The format of a link name is:

/dev/rst[0-127]

The link name does not contain the **b** character. The link name points to the actual BSD device name. For example, link name /dev/rst12 may point to the device file name /dev/rmt/0mb and the link name /dev/rst20 may point to the device file name /dev/rmt/0hb.

You must specify the actual name of either the BSD or non-BSD tape device in your EBR script. EBR takes the appropriate action based on the type of device driver. If you specify a link name (for example, /dev/rst12), EBR issues an extensive error message when it attempts to open the tape for a read or write operation. The error message sup-

plies detailed syntax information for specifying tape device names in the Solaris environment.

Use the **man mtio** command to see the manual pages for MTIO, which includes detailed information on BSD and non-BSD behavior.

Retaining Tapes

Before you use EBR, decide how long you want to retain your backup tapes before reusing them and gather information required to prelabel the backup tapes you plan to use.

EBR maintains a backup count in the tape label that you can use to determine when to retire a tape from service (for example, before the tape media degrades from extensive use).

Selecting Tape Identifiers

Before you use a tape for an EBR backup, you must prelabel the tape with the EBR_label utility. Before you can run EBR_label, you must choose two identifiers for each tape cartridge: a tape serial number and a volume group name.

See **“Appendix A – Programs and Utilities” on page 265** for details on how to use the EBR_label utility. For details on selecting tape serial numbers and volume group names, see **“Choosing a Tape Serial Number” on page 144** and **“Choosing a Volume Group Name” on page 145**.

Determining the Number of Tape Cartridges

Full backups may require more cartridges than interval backups. The amount of data backed up by an interval backup is only a fraction of

that backed up by a full backup. The fraction increases as the number of days since the last full backup increases.

For example, you keep all your backup tapes for the last four weeks before recycling the tape cartridges. You make full backups on Sunday and one interval backup per day for the other 6 days of the week. In 4 weeks, you accumulate 4 full backups plus 24 interval backups (6 interval backups per week times 4 weeks). You must prelabel all 28 tapes to be used for the four-week period.

To determine the number of tapes required for all the systems you wish to back up, you need to know:

- Approximately how much data is in use for each magnetic disk dataset
- The average compression factor for the in-use data of each dataset type to be backed up
- How much in-use magnetic disk data you need to back up for each interval backup
- The data capacity of your tape cartridge

Compression Factors

EBR always unconditionally compresses data before writing it to the tape drive. EBR applies a different compression factor for each data type, as shown in the table below.

EBR Compression Factors

Data Type	Typical Compression Factor *
Page cache object	1.0 up to 1.25 Most page cache objects are compressed in storage. **EBR may uncompress some of those images due to error checking**.
MKF data	0.625
Oracle data	0.34

* Actual compression factors vary depending on the data.

For example, EBR applies its compression factor to MKF data in the following manner:

100 MB MKF data * 0.625 = 62.5 MB

One hundred megabytes of magnetic disk-resident MKF data is stored in 62.5 megabytes of space on the backup tape.

Note The compression factors take into account additional error checking and correction (ECC) blocks written to tape by EBR.

Each database contains blocks that are in use (contain data) and unused blocks. A new database typically has long sequences of unused (all-zero) blocks at the end of the database. New rows of data are inserted into these unused blocks and become “in-use” blocks.

EBR uses different compression factors to compress in-use and unused blocks. Unused blocks are compressed at a much higher ratio than in-use blocks. When calculating how much data you have to back up, always assume the worst case—all blocks are in use. The EBR compression factors you use in your initial calculations will result in very conservative estimates because of the large number of all-zero blocks at the end of the databases. However, as the databases fill, the number of used blocks increases. Because your initial assumption (all blocks are in use) overestimated the number of prelabeled tapes necessary to hold the backup data, you will have room to grow as the amount of in-use data increases. You will not have to add tapes to the volume group later as the databases fill.

Determining the Amount of Changed Data for Interval Backups

Only a fraction of database blocks change in a day and only experience can provide the appropriate percentage factor to apply. Initially assume that ten percent of the in-use blocks of the database are modified each day. Ten percent is probably an overestimate. Examine the daily interval backup progress logs to see how many blocks are actually written to tape each day and plan accordingly.

Do not worry about inaccurate initial estimates. The worst that can happen is that the data does not fit on the tape and the backup terminates unsuccessfully. If that happens, break large datasets into parts and add more threads to the backup script. Then perform another backup. Monitor backup progress logs. In particular, examine the total number of compressed blocks written to tape and compare this value to the tape cartridge capacity. When the value is close to the cartridge capacity, add another thread to your scripts.

Tape Cartridge Capacity

When determining how much data fits on a tape cartridge, use no more than 80% of the cartridge capacity. This percentage allows for the inherent variability of the compression factors from one data type to another.

Even if your tape drive includes a hardware data compression feature, use the **uncompressed** capacity of the cartridge to determine how much data fits on one cartridge. EBR unconditionally compresses data before writing it to the tape drive so it does not matter whether the compression hardware of the tape drive is enabled or disabled: data that is already compressed generally does not compress much more.

Note Additional data compression may be available by enabling a tape drive hardware compression feature. However, We recommend that you turn off hardware compression for your tape drives. See **"Data Compression" on page 58** for more information.

Note Estimate that an additional 25% of space will be required for the page cache in a full backup scenario. For example, a 2GB page cache may require 2.5GB on tape due to error checking and correction blocks (ECC).

EBR supports the tape types listed in the following table. The tape length in meters, the uncompressed capacity, and an 80% capacity value are also listed to help you understand how much data fits on one cartridge based on capacity used. See **"Tape Support" on page 33**

for more information about the tape device types supported and on which platforms.

Tape Cartridge Capacity

Tape Type (Length in Meters)	Uncompressed Capacity	80% of Capacity
8mm (15 m)	933 P E	7; 3 P E
8mm (54 m)	518 J E	5 J E
8mm (112 m)	718 J E	619 J E
8mm (160 m)	: J E	819 J E
8mm “Mammoth” (170 m)	53 J E	49 J E
4mm (60 m)	415 J E	<93 P E
4mm (90 m)	5 J E	419 J E
QIC	833 P E	733 P E

Tape Cartridge Calculation Example

The following basic steps are required to estimate the number of prelabeled tape cartridges you need to perform a backup:

- Determine the uncompressed size of datasets to be backed up.
- Calculate the compressed size of each dataset to be backed up.
- Calculate the number of tapes to prelabel.

The following procedure estimates the number of tape cartridges required to perform a full backup of page cache, three MKF databases, and the Oracle index database.

Note Dataset names and sizes are examples. Your dataset names and sizes may vary.

1. Start the Configuration Editor.
   - On Windows Server systems, double-click the Configuration Editor icon.
   - On UNIX systems, enter **fn_edit** at the command line.
2. Click on the Datasets tab.
3. Fill in the size of each dataset in the table below (example sizes are in bold).

Data Type	Dataset	Maximum Size (MB)
Page Cache	cache0	**400**
MKF	permanent_db0	**400**
	transient_db0	**100**
	sec_db0	**20**
Oracle	oracle_db0	**2000**

4. Estimate the compressed size of each dataset using the following formula:

   total uncompressed size * compression factor = total compressed size

Assume all blocks are in use and complete the following table (example sizes are in bold).

Data Type (Dataset)	Total Uncompressed Size (MB)	Compression Factor	Total Compressed Size (MB)
Page Cache (cache0)	400	1.25	**500**
MKF (permanent_db0)	400	0.625	**250**
MKF (transient_db0)	100	0.625	**62.5**
MKF (sec_db0)	20	0.625	**12.5**
Oracle (oracle_db0)	2000	0.34	**680**

5 Calculate the number of tape cartridges.

This example uses 8mm tape, 54 meters in length. From the table on **page 141**, you know that 80% of this tape's capacity is 2000 MB. The backup totals 1505 MB of compressed data so you only need to prelabel one tape cartridge to hold this backup data.

Your datasets may exceed the capacity of your tape cartridge. If so, calculate the number of tapes you need and determine which datasets fit on a particular cartridge. Then code the thread assignments in your backup script accordingly. Remember that order constraints affect some dataset combinations.

Tip When individual datasets are too large to fit on a single tape cartridge, break the dataset into multiple parts. In your script, assign each part of the dataset to a separate cartridge. EBR then backs up this single large dataset to multiple tape cartridges. See **"Threads" on page 49** for more information and the example in **"Special Considerations for Listing Striped Datasets" on page 211**.

Choosing a Tape Serial Number

The tape serial number is an identifier. When choosing an identifier, adhere to the following restrictions:

- A tape serial number must consist only of uppercase ASCII letters, ASCII decimal digits, and the ASCII underscore character.
- The first character of a tape serial number must be an uppercase letter.
- The maximum number of characters in a tape serial number is 12; the minimum number of characters is 1.

We recommend that you write the tape's volume group name and tape serial number on a human-readable external label affixed to the tape cartridge.

Each tape in your enterprise should have a unique tape serial number. Develop a systematic scheme for assigning tape serial numbers that will help you determine to what system the tape belongs. You can also use the scheme to identify other information you may consider important about the tape. For example, a numbering scheme that indicates when the tape was placed in service is important for determining when the tape should be taken out of service (tapes eventually wear out).

One way to assign unique tape serial numbers is to use an abbreviation of the system name as the first part of the tape serial number, followed by the year, month, and day the tape was placed in service, and a sequential number for that day. For example, using the abbreviation "BDT" for a system named bandit, the first tape labeled on July 9, 2004, has a tape serial number of BDT040709001. After initially agreeing on the abbreviations for system names, system administrators across your enterprise can then assign tape serial numbers

without coordinating with each other. Anyone can determine which system the tape is for and the tape's creation date by examining the tape serial number.

Choosing a Volume Group Name

Every tape used in a backup must have the same volume group name and a different tape serial number. We recommend that each different backup have a different volume group name.

The restrictions on volume group names are as follows:

- Volume group names must consist only of uppercase ASCII letters, ASCII decimal digits, and the ASCII underscore character.
- The first character of the volume group name must be an uppercase letter.
- The maximum number of characters in a volume group name is 28; the minimum number of characters is 1.

Adopt a convention for volume group names that indicates the most relevant information about the volume group. For example:

- The system backed up
- Day of the week
- Morning (AM) or evening (PM)
- Cycle number (if you back up on the recommended weekly cycle and keep four weeks of tapes, cycle number is a value from 1 to 4)
- Online or offline backup
- Full or interval backup

For example, from the BANDIT_MONPM3_ONLIVAL volume group name, you can determine that an online interval backup was performed on Monday evening of cycle number 3 for a system called bandit.

Organizing Your Backup Tapes

When storing your tapes, consider physically organizing your tapes in a tape rack by volume group name within system name as opposed to simply organizing by tape serial number. Then all the tapes you need for a backup or restore are in one place instead of scattered throughout the rack.

5
Developing Your Scripts

This chapter provides you with script planning and implementation guidelines and script syntax.

Planning Your Scripts

Before you write your backup and restore scripts, read **Chapter 4, "Developing Your Backup Strategy," on page 89**. Implement your backup strategy in your scripts. EBR provides a set of sample scripts, some that you can use as is and others you can edit to meet your needs. Sample scripts include, but are not limited to, those for backing up and restoring the following systems:

- Combined server systems
- Oracle index database systems

Additional customized scripts can be easily generated using the EBR_genscript tool. (See **"EBR_genscript" on page 268**.)

Note We recommend that you create a directory and copy all FileNet-provided scripts into your directory. Edit and test the script copies in your directory. New sample scripts may be added and existing sample scripts may be updated occasionally. During FileNet software installations or upgrades, the new sample scripts are copied into the FileNet directories listed below, replacing existing scripts. Therefore, the

scripts you have modified and use regularly should be in your own separate directory.

In addition, protect your customized scripts by backing them up to tape with a file copy utility. You may also want to have a printed copy available.

The sample scripts are located in the following directories:

/fnsw/local/EBR/samples	For UNIX systems
\fnsw_loc\EBR\samples (or any directory of your choice)	For Windows Server systems

A READ_ME file in the directory explains each sample script and its use.

Note Descriptions of each sample script are in **"Appendix J – Restoring Oracle" on page 411**. EBR scripts conform to the Backus-Naur Form script specification, which is included in Appendix I.

The samples directory also contains two template files:

- template.bac describes all required and optional EBR backup statements
- template.res describes all required and optional EBR restore statements

For information about each template file, see **"Template File Descriptions" on page 161**.

Before you begin to develop scripts for backup and restore, you should complete the following tasks:

1. Decide on a centralized, decentralized, or mixed strategy.
2. Decide which datasets to back up.
3. Decide on offline or online backups.
4. Determine if you need to share tape drives between threads.
5. Decide whether to back up and restore using disk files or tape.
6. Develop a scheme for assigning tape serial numbers and volume group names.
7. Determine a retention period for backup tapes.
8. Prelabel all tapes using EBR_label.

Steps 4 and 5 above are discussed in the following topics. All other steps are discussed in **Chapter 4, "Developing Your Backup Strategy," on page 89**.

Sharing Tape Drives Among Threads

As you plan your scripts, consider whether to share tape drives between threads.

EBR does not support overflow tapes but you can approximate this concept by sharing tape drives between threads. To simulate overflow tapes, the backup and restore scripts must specify at least two threads and that at least one drive is shared by multiple threads. See **"Data Overflow" on page 129** for more information.

You can also use shared tape drives when the number of available operative tape drives is less than the number of threads. Sharing drives makes the backup or restore possible under this circumstance.

To share a tape drive among threads, specify the same tape drive in one or more threads in the backup_device statement of the THREADS section of your script. All cartridges must be compatible with the drive. At run time, the threads gain access to the drive in increasing thread number order.

Backing Up to and Restoring from Magnetic Disk Files

EBR supports backing up to and restoring from a magnetic disk file. Backup disk files are treated like tapes.

Note Consider backing up to magnetic disk files when debugging your backup scripts, training personnel, or if performance is not adequate when using the Exabyte Model 8900 tape drive (see **"Using the Exabyte Model 8900 ("Mammoth") Tape Drive" on page 134**). You can also use magnetic disk file backups as an additional safeguard when using a third party product to back up your system. If you choose to do your production backups to magnetic disk files, be sure to do a secondary backup of the magnetic disk file to tape.

A magnetic disk file is similar to a tape file in the following ways:

- You must prelabel the disk file. Prelabeling a disk file automatically creates the file.
- A disk file cannot be overwritten by another backup until the backup expires.
- You can run EBR_label to erase the data to make the file available for another backup before the backup expires.
- You can run EBR_tdir to display the volume label.

If you back up to tapes, you must restore from those tapes. You cannot restore from a copy of those tapes. If you back up to a disk file, you must restore from that disk file, not from a copy of that disk file. One reason for this restriction is that on tape, tape marks are a special type of information whereas in disk files, tape marks are just another block type. If a tape drive is equipped with a high-speed search capability, EBR uses that capability to search for tape marks. If you copy a backup disk file to tape (for example, with the UNIX dd program), the tape marks written to tape are recognized as just another regular tape block. Therefore, a high-speed search on tape skips right over your dataset and EBR will not find it.

For the same reason, if you use a utility (for example, the UNIX dd program) to copy an EBR-created backup tape to a magnetic disk file, you will not be able to use the copied disk file to restore your datasets. For more information about EBR tape format, see **"Appendix B – Tape Format" on page 360**.

Script Syntax Guidelines

Scripts are ASCII files. You can create and edit them with a text editor. However, you must adhere to a set of syntax guidelines as described in this section. Correct syntax is automatically generated if you choose the EBR_genscript tool to create your dataset definitions file, device specifications files, and your backup and restore scripts. Although syntax is described in detail in this section, use of EBR_genscript is recommended and assumed.

Tokens

Within a script you use syntax tokens such as identifiers, decimal integers, strings enclosed in double quotes, and special characters such as equal sign (=), semicolon (;), and comma (,).

For example, the names you assign to datasets in your script are identifiers. Identifiers must start with a letter and can consist of any combination of uppercase and lowercase ASCII letters, ASCII decimal digits, and the ASCII underscore character (_). No embedded blanks are allowed. An identifier must be less than 80 characters long.

Parameters

Scripts can have parameters. In the script, parameter variables are identifiers prefixed with a dollar sign ($). If you use parameters in your script, you must specify the parameter variable on the command line at run time. On the command line, you specify the value of a parameter as:

<parameter_variable_name>=<value>

EBR resolves the parameter variable in your script at run time by substituting the value you enter on the command line for the parameter variable in your script.

Note When you enter a variable on the **command line**, do not include the leading dollar sign ($) and do not insert spaces on either side of an equal sign (=).

You may find it useful to specify tape serial numbers and volume group names as parameters. This is especially helpful if you back up the

same datasets every day, but you use a different set of tapes every day, and the rest of your script stays the same.

Suppose you have the following line in a script:

```
volume_serial = $ser;
```

To substitute BDT960409001 for $ser, the command line parameter is:

```
ser=BDT960409001
```

When you run the backup or restore script, the substitution process changes the line in your script to:

```
volume_serial = BDT960409001;
```

Because the UNIX shell creates separate parameters for each embedded blank encountered, embedded blanks are not allowed in an EBR command line parameter and, if found, generate errors. For example, compare the following parameter specifications for embedded blanks before and after the equal sign (=):

`ser = BDT960409001`	incorrect specification
`ser=BDT960409001`	correct specification

Aligning Text

To vertically align text in your scripts, use blank characters. Do **not** use the tab key to align text. EBR generates a syntax error if it encounters a tab character in your script.

Specifying Network Host Locations

In your scripts, you must provide host location information. For example, in the DATASETS section, you provide the location of the hosts for the magnetic disk datasets you wish to back up. In the DEVICE_SPECIFICATIONS sections, you provide the host locations to which the tape drives are connected. You can specify the location of a host in one of four formats:

- Three-part NCH names

  Format: “<object_name>:<domain_name>:<enterprise_name>”

  Example: “TapeServer1:SystemB:FileNet”

- NCH domain name

  Format: “<domain_name>:<enterprise_name>”

  Example: “SystemB:FileNet”

- TCP/IP address

  Example: “135.0.2.112”

- Host name

  Example: “costa2”

The name or IP address must be enclosed in double quotes for any format you choose.

Script Sections

An EBR script has the following sections, in order:

- Script format level specification
- BACKUP_GLOBAL_PARAMETERS or RESTORE_GLOBAL_PARAMETERS, depending on script type
- DATASETS
- DEVICE_SPECIFICATION
- BACKUP_OPTIONS or RESTORE_OPTIONS, depending on script type
- THREADS

Each section is briefly described below. Details of how to write each section for both backup scripts and restore scripts are described later in this chapter.

With the exception of the script format level identifier, the beginning and ending keywords of each major section must be in **uppercase**.

The general form of a backup script is shown below:

```
EBR_script( format_level = 2; );
BACKUP_GLOBAL_PARAMETERS
     ...
END_BACKUP_GLOBAL_PARAMETERS
DATASETS
     ...
END_DATASETS
DEVICE_SPECIFICATION
     ...
END_DEVICE_SPECIFICATION
BACKUP_OPTIONS
     ...
END_BACKUP_OPTIONS
THREADS
     ...
END_THREADS
```

Note The DATASETS section can be replaced with an **include** statement that references a file (datasets.inc) that describes the datasets.

The DEVICE_SPECIFICATIONS can be replaced with an **include** statement that represents a file (device.dev) that describes the backup devices.

The general form of a restore script is shown below:

```
EBR_script( format_level = 2; );
RESTORE_GLOBAL_PARAMETERS
     ...
END_RESTORE_GLOBAL_PARAMETERS
DATASETS
     ...
END_DATASETS
DEVICE_SPECIFICATION
     ...
END_DEVICE_SPECIFICATION
RESTORE_OPTIONS
     ...
END_RESTORE_OPTIONS
THREADS
     ...
END_THREADS
```

Note The DATASETS section can be replaced with an **include** statement that references a file (datasets.inc) that describes the datasets.

The DEVICE_SPECIFICATIONS can be replaced with an **include** statement that represents a file (device.dev) that describes the backup devices.

The parallel construction between backup and restore scripts helps you develop your scripts. Using a text editor, you can develop a restore script by editing a backup script or vice versa.

Script Syntax Format Level

The first line of an EBR script identifies the script syntax format level, which enables EBR to recognize different script releases. If changes to

the script syntax are made in the future, EBR can identify the format level at run time and read the script properly whether it is written in the new or old syntax. For example, EBR for Image Services 3.3.1 uses script format level 1 while EBR for Image Services 3.4.0 uses script format level 2. EBR for IMS 3.4.0 is able to read scripts in either format, however, **you cannot mix the two formats in a single script**. The syntax of the script format level statement is:

```
EBR_script ( format_level = 2; );
```

GLOBAL_PARAMETERS Section

The BACKUP_GLOBAL_PARAMETERS or, for restore scripts, the RESTORE_GLOBAL_PARAMETERS section describes the options that globally affect the backup or restore.

DATASETS Section

The DATASETS section describes each dataset to be backed up or restored. Since the information needed for the DATASETS section is identical for both backup and restore scripts, you can create a single ASCII text file that contains a list of all the datasets you want to back up or restore. Then use the EBR include feature in the script language to include the ASCII file. To call out or reference an inclusion file, insert a **#** character in the first column of your script followed immediately by the keyword **include** and the full path name of the text file, surrounded by double quotes, to be included. The example below causes a file named /fnsw/local/EBR/datasets.scr to be included in the script:

```
#include “/fnsw/local/EBR/datasets.scr”
```

The /fnsw/local/EBR/samples directory contains a sample include file, datasets.inc. Some of the sample scripts use this include file. You can

modify a copy of the datasets.inc file for use in developing your own scripts.

You can also use the EBR_genscript tool to create the include file. From the EBR_genscript main menu, select “Generate dataset definitions” and enter a file name at the prompt, as shown in the following example:

```
===== EBR_genscript Main Menu =====

  [1] - Generate dataset definitions
  [2] - Generate Device Specification
  [3] - Generate backup script
  [4] - Generate restore script
  [9] - Help
  [0] - Exit

Enter command ==> 0

Enter EBR dataset definition file name
(suffix '.ddf' will be appended to the file name
entered)

==> /fnsw/local/EBR/tt
```

The file name can be any name you choose. EBR_genscript appends the default file extension of .ddf to your file name. If you do not specify a full path name, EBR_genscript creates the file in the current directory. EBR_genscript then issues a series of prompts to begin the dataset definition process. Your responses to these prompts cause the entries to be dynamically added to the .ddf file. For more information, see **“EBR_genscript” on page 268**.

DEVICE_SPECIFICATION Section

The DEVICE_SPECIFICATION section allows you to define the backup or restore device, such as a disk file, tape in a standalone tape drive, or tape in a tape library.

Since the information needed for the DEVICE_SPECIFICATION section is identical for both backup and restore scripts, you can also create a single ASCII text file that contains the descriptions of the devices you will be using. Then use the EBR include feature in the script language to include the ASCII file. To perform the inclusion of the file, insert a **#** character in the first column of your script followed immediately by the keyword **include** and the full path name of the text file, surrounded by double quotes, to be included. The example below causes a file named /fnsw/local/EBR/device.dev to be included in the script:

#include “/fnsw/local/EBR/devices.dev”

BACKUP_OPTIONS or RESTORE_OPTIONS Section

The BACKUP_OPTIONS section or, for a restore script, RESTORE_OPTIONS section, follows the DATASETS section and describes the type of backup or restore to perform on each dataset.

THREADS Section

The THREADS section describes each thread used in the backup or restore. Because a one-to-one relationship exists between a thread and a cartridge, this section has one thread description per tape cartridge used in the backup or restore. It also identifies the type of device as specified in the DEVICE_SPECIFICATION section for each thread.

Template File Descriptions

template files are provided to help you understand the statement and parameter requirements of backup and restore scripts. The backup template file, template.bac, describes each backup statement along with information about the statement's requirements, syntax information, and defaults. The restore template file, template.res, describes similar information for restore. You can find these template files in the EBR samples directory:

UNIX /fnsw/local/EBR/samples For UNIX systems

WIN <drive>:\fnsw_loc\EBR\samples
(or any directory of your choice) For Windows Server systems

In the sections below, tables summarize the information found in the template files. You can use the tables as a quick reference to information as you write your scripts. The actual template file follows the table of information.

Device Statements Table

Use the Device Statements table as a quick reference to the statements that create a tape, tape library, or disk file as the backup or restore device.

Note Information in the table columns adheres to case sensitivity requirements for section names and dataset types.

The table identifies the name for each section and subsections of the script, the statements in the section or subsection, whether or not the

statement is required, and a short description including any default value.

Refer to the contents of the template.bac and template.res files in the sections, **"Backup Statements Table" on page 163** and **"Restore Statements Table" on page 171**, for examples of device statement output. .

Device Statements Table

Section	Subsections	Statement	Descr
DEVICE_ SPECIFICATIONS			
	disk_file1	location	Required. Hos disk file.
		filename	Required. File file.
	tape_drive	location	Required. Hos tape drive. On per thread.
		drive_name_rewind	Required if bac tape or tape lib name of a rew vice. One occu thread.
		drive_name_no_rewind	Required if bac tape or tape lib name of a non- device. One oc thread.

Device Statements Table, Continued

Section	Subsections	Statement	Descr
		drive_type	Required if bac tape or tape lib 8mm, 4mm, DA One occurrenc
	tape_library	location	Required. Hos tape drive in ta occurrence pe
		library_device	Required. Full tape library.
		library_type	Required. Spe (8mm), “EXB-2 “EXB-220” (ma
	tape_drive	num_drives	Required. Num drives in tape l
		drive_name_rewind	Required. Spe rewinding tape occurrence pe
		drive_name_no_rewind	Required. Spe non-rewinding One occurrenc
		drive_type	Required. Spe 4mm, DAT, or currence per th

Backup Statements Table

Use the Backup Statements table as a quick reference to the statements that create backup scripts. Information in the table corresponds to the template.bac file, which follows the table.

Note Information in the table columns adheres to case sensitivity requirements for section names and dataset types.

The table identifies each section of the script along with each statement by dataset type (if applicable), whether or not the statement is required, and a short description including any default value.

The contents that follow the Backup Statements table is the actual template.bac file. The template file in this manual may differ slightly from the template file on your system. Refer to the template file in the samples directory for the most current version of the template file.

Backup Statements Table

Section	Dataset Type or Name	Statement	Description
BACKUP_GLOBAL_PARAMETERS	N/A		Designates beginning of section.
		volume_group	Required. 1-28 characters.
		expiration	Required.
		start_date	Not required.
		start_time	Required only if you specify start_date.
DATASETS <dataset_name>:<dataset_type> end_<dataset_type>	partition	location	Required.
		filename	Required.
		start_block	Not required. Default=0
		block_size	Not required. Default=1024 bytes
		num_blocks	Not required. Default=all blocks of this partition
	MKF	location	Required.

Backup Statements Table, Continued

Section	Dataset Type or Name	Statement	Description
		base_data_file	Required.
		transient_db	Required for transient database only.
	Oracle	location	Required.
		signature_file_ directory	Required.
	cache	location	Required.
		transient_db	Required.
		permanent_db	Required.
		security_db	Required.
		order_constraints	Required only if backing up cache. Multiple statements are allowed in an order constraints paragraph. (See examples on **page 194**.) order_constraints must be the **last** subsection in DATASETS.
BACKUP_OPTIONS	partition	None	Not required.
	perma-nent	full_backup **or** interval_backup	Required (mutually exclusive).
		offline_backup **or** online_backup	Required (mutually exclusive).
	security	full_backup **or** interval_backup	Required (mutually exclusive).
		offline_backup **or** online_backup	Required (mutually exclusive).

Backup Statements Table, Continued

Section	Dataset Type or Name	Statement	Description
		offline_backup **or** online_backup	Required (mutually exclusive).
		archive_redo_ log_retention	Required. (minimum=2 days)
	cache	full_backup	Required. (Cache backup **must** be performed offline.)
THREADS		num_threads	Required.
		thread *n*	One per thread.
		device	Required. Specify if disk file, tape drive, or tape library.
		volume_serial	Required. 1-12 characters. One occurrence per thread.
		datasets <dataset list>*;*	Required. Each comma-separated element in the list is of the form <identifier> or "<identifier> <part *n* of *m*)" where <identifier> is the name assigned to a dataset in the DATASETS section.
		end_thread	Required.

The contents of the **template.bac** file are listed below:

```
-- template.bac
-- Purpose: This script describes all required and optional statements for
-- EBR backup.
```

```
EBR_script (format_level = 2;);

BACKUP_GLOBAL_PARAMETERS

   volume_group = TEST_VG;       -- required (all uppercase, 1-28 chars)
   expiration = 30 days;         -- required (day(s), week(s), month(s))
   start_date = 1999/12/31;      -- optional
   start_time = 05:00;           -- optional (required if start_date is
                                 -- specified)

END_BACKUP_GLOBAL_PARAMETERS

-- The DATASETS section is exactly the same for restore scripts as for
-- backup scripts.  Therefore, it is strongly recommended that the DATASETS
-- section be saved in a separate ASCII file, and that all backup and
-- restore scripts use the "#include" directive to include this file.

DATASETS

   part: partition
     location = "test_domain:FileNet";-- required; dataset host or NCH name
     filename = "/x/y";               -- required
     start_block = 0;                 -- optional (default=0)
     block_size = 1024 bytes;         -- optional (default=1024 bytes)
     num_blocks = 100;                -- optional (default=all blocks)
   end_partition

   sec: MKF
     location = "test_domain:FileNet";-- required; dataset host or NCH name
     base_data_file = "/fnsw/dev/1/sec_db0";      -- required
   end_MKF

   perm: MKF
     location = "test_domain:FileNet";-- required; dataset host or NCH name
     base_data_file = "/fnsw/dev/1/permanent_db0";     -- required
   end_MKF

   inxdb: Oracle
     location = "test_domain:FileNet";-- required; dataset host or NCH name
     signature_file_directory = "/fnsw/local/tmp/ora_sig"; -- required
   end_Oracle
```

```
    tran: MKF
      location = "test_domain:FileNet";   -- required; dataset host or NCH name
      base_data_file = "/fnsw/dev/1/transient_db0"; -- required
      transient_db;                                 -- required for transient database
    end_MKF


    cache: cache
      location = "page_cache1:test_domain:FileNet";-- required; dataset host or NCH
name
      transient_db = tran;                             -- required
      permanent_db = perm;                             -- required
      security_db = sec;                               -- required
    end_cache

    order_constraints                                  -- required only
      sec, perm, tran before cache;                    -- if transient
    end_order_constraints                              -- database and
                                                       -- cache are defined
 END_DATASETS

DEVICE SPECIFICATIONS
 TAPE_DEV1: tape_drive                    -- required if backup device is tape drive
   location = "test_domain:FileNet";                -- required
   drive_name_rewind = "/dev/rmt0";                 -- required
   drive_name_no_rewind = "/dev/rmt0.1";            -- required
   drive_type = 8mm;                                -- required
 end_tape_drive

 TAPE_DEV2: tape_drive                    -- required if backup device is tape drive
   location = "test_domain:FileNet";                -- required
   drive_name_rewind = "/dev/rmt1";                 -- required
   drive_name_no_rewind = "/dev/rmt1.1";            -- required
   drive_type = 8mm;                                -- required
 end_tape_drive

 DISK_FILE1: disk_file                    -- required if backup device is disk file
   location = "test_domain:FileNet";                -- required
   filename = "/fnsw/local/tmp/disk_backup";        -- required
---Description: "A disk file backup"                -- optional
```

```
 end_disk_file

 LIB1: tape_library                          --required if backup device is tape library
   location = "test_domain:FileNet";                  -- required
   library_device = "/dev/chgr0";                     -- required
   library_type = "EXB-210"                           -- required. Specify one.

   tape_drives
    num_drives = 1;                                    -- required
    drive 1
     drive_name_rewind = "/dev/rmt1";                  -- required
     drive_name_no_rewind = "/dev/rmt1.1";             -- required
     drive_type = 8mm;                                 -- required
   end_tape_drives
---Description: "Exabyte tape library"                -- optional
 end_tape_library

END_DEVICE_SPECIFICATION

 BACKUP_OPTIONS

    part: backup_options    -- No backup options exist for partition.
    end_backup_options      -- These two lines can be omitted or specified.

    sec: backup_options

        full_backup;        -- required; specify either "full_backup" or
                            --           "interval_backup".

        offline_backup;     -- required; specify either "online_backup" or
                            --           "offline_backup".

    end_backup_options

    perm: backup_options

        full_backup;        -- required; specify either "full_backup" or
                            --           "interval_backup".

        offline_backup;     -- required; specify either "online_backup" or
                            --           "offline_backup".
```

```
    end_backup_options

    inxdb: backup_options

        full_backup;       -- required; specify either "full_backup" or
                           --           "interval_backup".

        offline_backup;    -- required; specify either "online_backup" or
                           -- "offline_backup".

        archive_redo_log_retention = 30 days;     -- required (minimum=2 days)

    end_backup_options


    tran: backup_options

        full_backup;     -- optional; only "full_backup" is supported
                         -- for transient database in the
                         -- current release.
        offline_backup;  -- optional; only "offline_backup" is supported
                         -- for transient database in the
                         -- current release.
    end_backup_options

    cache: backup_options

        full_backup;     -- required; only "full_backup" is supported
                         -- for cache in the current release

    end_backup_options

END_BACKUP_OPTIONS

THREADS

 num_threads = 3;       -- required

 thread 1
   device = TAPE_DEV1;
```

```
    volume_serial = TAPE_SER1;     -- required (1-12 chars, all uppercase)
    datasets                       -- required
        perm (part 1 of 3);        -- specific datasets for backup on this thread
  end_thread   --thread 1--

  thread 2
    device = LIB1(drive 1);
    volume_serial = BAR_0002000;    -- tape serial number format if tape lib is
                                           -- barcode labeled
    datasets                         -- required
       part, sec, perm (part 2 of 3),       -- datasets for backup; if performing
                                            -- a backup of the tran, cache,
                                            -- cache, a backup of its transient
                                            -- database is required.
       inxdb (part 1 of 2);           -- (part <n> of <m>) is required for
                                      --  striped datasets.
  end_thread   --thread 2--

  thread 3
    device = TAPE_DEV2;
    volume_serial = TAPE_SER2;   -- tape serial number (1-12 chars, all uppercase)
    datasets
        part, sec, perm, (part 3 of 3),   -- datasets for backup; if performing
                                          -- a backup of the tran, cache,
                                          -- cache, a backup of its transient
                                          -- database is required.
       inxdb (part 2 of 2);        -- (part <n> of <m>) is required for
                                   -- striped datasets.
  end_thread   --thread 3--

END_THREADS
```

Restore Statements Table

Use the Restore Statements table below as a quick reference to the statements you need to create backup scripts. Information in the table corresponds to the template.res file.

Note Information in the table columns adheres to case sensitivity requirements for section names and dataset types.

The tables identify the beginning and ending name for each section of the script along with each statement by dataset type (if applicable), whether or not the statement is required, and a short description including any default value.

The template file in this manual may differ slightly from the template file on your system. Refer to the template file in the samples directory for the most current version of the template file.

The Restore Statements table below is followed by the contents of the actual template.res file.

Restore Statements

Section	Dataset Type or Name	Option	Description
RESTORE_GLOBAL_PARAMETERS	N/A	volume_group	Required. 1-28 characters
		tape_mount_timeout	Optional. The default is no timeout.
DATASETS <dataset_name>:<dataset_type> end_<dataset_type> ...	partition	location	Required.

Restore Statements, Continued

Section	Dataset Type or Name	Option	Description
		filename	Required.
		start_block	Not required. Default=0
		block_size	Not required. Default=1024 bytes
		num_blocks	Not required. Default=all blocks of this partition
	MKF	location	Required.
		base_data_file	Required.
		transient_db	Required for transient database only.
	Oracle	location	Required.
		signature_file_ directory	Required.
		parameter_file	Required only if Oracle parameter file is other than the default, /fnsw/local/oracle/ init.ora.
DATASETS (continued)	cache	location	Required.
		transient_db	Required.
		permanent_db	Required.
		security_db	Required.

Restore Statements, Continued

Section	Dataset Type or Name	Option	Description
	N/A	order_constraints	Required only if restoring cache. Multiple statements allowed in an order_constraints paragraph. (See examples on **page 194**.) order_constraints must be the **last** statement in DATASETS.
RESTORE_OPTIONS	partition	restore_onto	Not required.
	permanent	full_restore **or** interval_restore	Required (mutually exclusive).
		interval_restore_ follows = {true | false}	Required if specifying full_ restore.
		reconfigure_onto	Not required; applies only when restoring to a new database and you specify full_ restore **and** interval_ restore_follows = false.
	transient	full_restore interval_restore_ follows=false	If restoring transient database, both statements are required under tran: restore_options.

Restore Statements, Continued

Section	Dataset Type or Name	Option	Description
RESTORE_OPTIONS (continued)	Oracle	full_restore **or** interval_restore	Required (mutually exclusive).
		interval_restore_ follows = {true | false}	Required if specifying full_ restore.
		restore_redo_ logs = {true | false}	Not required; use only for unusual circumstances. See **"Help for Unusual Circumstances" on page 227**.
		restore_control_file = {true | false}	Not required; use only for unusual circumstances. See **"Help for Unusual Circumstances" on page 227**.
		rollforward = {true | false}	Not required; use only for unusual circumstances. See **"Help for Unusual Circumstances" on page 227**.
	cache	full_restore	Required.
		interval_restore_ follows = false	Required. Always set to false.

Restore Statements, Continued

Section	Dataset Type or Name	Option	Description
THREADS	N/A	num_threads	Required.
		thread *n*	One per thread
		device	Required (specify tape, tape library, or disk file). One occurrence per thread.
		volume_serial	Required. 1-12 characters. One occurrence per thread.
		datasets	Required.
		end_thread	Required.

The contents of **template.res** are shown below.

```
-- template.res
-- Purpose: This script describes all required and optional statements for
-- EBR restore.
EBR_script (format_level = 2;);

RESTORE_GLOBAL_PARAMETERS

   volume_group = TEST_VG;     -- required (all uppercase, 1-28 chars)
END_RESTORE_GLOBAL_PARAMETERS
```

```
-- The DATASETS section is exactly the same for restore scripts as for
-- backup scripts.  Therefore, it is strongly recommended that the DATASETS
-- section be saved in a separate ASCII file, and that all backup and
-- restore scripts use the "#include" directive to include this file.

DATASETS

   part: partition
     location = "test_domain:FileNet";        -- required; dataset host or NCH name
     filename = "/x/y";          -- required
     start_block = 0;            -- optional (default=0)
     block_size = 1024 bytes;    -- optional (default=1024 bytes)
     num_blocks = 100;           -- optional (default=all blocks)
   end_partition

   sec: MKF
     location = "test_domain:FileNet";         -- required; dataset host or NCH name
     base_data_file = "/fnsw/dev/1/sec_db0";   -- required
   end_MKF

   perm: MKF
     location = "test_domain:FileNet";               -- required; dataset host or NCH name
     base_data_file = "/fnsw/dev/1/permanent_db0";   -- required
   end_MKF

   inxdb: Oracle
     location = "test_domain:FileNet";                -- required; dataset host or NCH name
     signature_file_directory = "/fnsw/local/tmp/ora_sig"; -- required
   end_Oracle
   tran: MKF
     location = "test_domain:FileNet";                -- required; dataset host or NCH name
     base_data_file = "/fnsw/dev/1/transient_db0";    -- required
     transient_db;                                    -- required for transient database
   end_MKF

   cache: cache
     location = "page_cache1:test_domain:FileNet";     -- required; dataset host or NCH name
```

```
      transient_db = tran;                          -- required
      permanent_db = perm;                          -- required
      security_db = sec;                            -- required
    end_cache

    order_constraints                             -- required only
      sec, perm, tran before cache;               -- if transient
    end_order_constraints                         -- database and
                                                  -- cache are defined
 END_DATASETS

DEVICE_SPECIFICATION
 TAPE_DEV1: tape_drive                          -- required if backup device is tape drive
   location = "test_domain:FileNet";            -- required; dataset host or NCH name
   drive_name_rewind = "/dev/rmt0";             -- required
   drive_name_no_rewind = "/dev/rmt0.1";        -- required
   drive_type = 8mm;                            -- required
 end_tape_drive

 TAPE_DEV2: tape_drive                          -- required if backup device is tape drive
   location = "test_domain:FileNet";            -- required; dataset host or NCH name
   drive_name_rewind = "/dev/rmt1";             -- required
   drive_name_no_rewind = "/dev/rmt1.1";        -- required
   drive_type = 8mm;                            -- required
 end_tape_drive

 DISK_FILE1: disk_file                          -- required if backup device is disk file
   location = "test_domain:FileNet";            -- required; dataset host or NCH name
   filename = "/fnsw/local/tmp/disk_backup";    -- required
---Description: "A disk file backup"            -- optional
 end_disk_file

 LIB1: tape_library                            --required if backup device is tape library
   location = "test_domain:FileNet";              -- required; dataset host or NCH name
   library_device = "/dev/chgr0";                 -- required
   library_type = "EXB-210"                     -- required.
```

```
   tape_drives
    num_drives = 1;                                        -- required
    drive 1
     drive_name_rewind = "/dev/rmt1";                      -- required
     drive_name_no_rewind = "/dev/rmt1.1";                   -- required
     drive_type = 8mm;                                     -- required
   end_tape_drive
---Description: "Exabyte tape library"                     -- optional
 end_tape_library

END_DEVICE_SPECIFICATION

 RESTORE_OPTIONS

    part: restore_options
        restore_onto "/x/y";                               -- optional
    end_restore_options


    perm: restore_options
        interval_restore;                                  -- optional
    end_restore_options

 -- or
 -- perm:
 --     full_restore;
 --     interval_restore_follows = false          -- required
 --     reconfigure_onto "/tmp/new_database";     -- optional; only applies when
 --                                                 -- performing a full restore
 --                                                 -- and no interval restore
 --                                                 -- follows.
 -- end_restore_options
 --
 -- or
 -- perm: restore_options
 --     full_restore;
 --     interval_restore_follows = true;           -- required
 -- end_restore_options
```

```
--
   inxdb: restore_options
       interval_restore;
   end_restore_options

-- or
-- inxdb: restore_options
--     full_restore;
--     interval_restore_follows = false;
-- end_restore_options
--
-- or
-- inxdb: restore_options
--     full_restore;
--     interval_restore_follows = true;
-- end_restore_options

-- For Oracle restore, the following 2 options are available to help deal with very unusual
-- circumstances, such as when not only the data file is bad, but control files or redo logs
-- are also corrupted. Normally, these options should NOT be specified.
--
-- Note that if option "restore_redo_logs" is set to true, "restore_control_file" should also
-- be set to true.

--          restore_control_file = true;              (default = false)
--          restore_redo_logs = true;                 (default = false)
--          rollforward = true;                       (default = true)

   tran: restore_options
       full_restore;
       interval_restore_follows = false;
   end_restore_options

   cache: restore_options
       full_restore;                      -- required; only "full_restore"
                                          -- is supported for cache in the current release
```

```
       interval_restore_follows = false;     -- required; always set to false for
       end_restore_options                    -- cache in the current release
 END_RESTORE_OPTIONS
 THREADS

  num_threads = 3;                           -- required

  thread 1
    device = TAPE_DEV1;
    volume_serial = TAPE_SER1;               -- tape serial number (1-12 chars, all uppercase)
    datasets
        perm (part 1 of 3);                  -- datasets to back up
                                             -- (part <n> of <m> is required for stripped sets
  end_thread

  thread 2
    device = LIB1(drive 1);
    volume_serial = BAR_0002000;    -- tape serial number format if tape lib is barcode labeled
    datasets
        part, sec, perm, (part 2 of 3),-- datasets for backup; if performing a backup of the
        tran, cache,                 -- cache, a backup of its transient database is required.
       inxdb (part 1 of 2);         -- (part <n> of <m>) is required for striped datasets.

  end_thread   --thread 2--

  thread 3
    device =  TAPE_DEV2;
    volume_serial = TAPE_SER2;            -- tape serial number (1-12 chars, all uppercase)
    datasets
        part, sec, perm, (part 3 of 3)  - datasets for backup; if performing a backup of the
        tran, cache,                 -- cache, a backup of its transient database is required.
       inxdb (part 2 of 2);         -- (part <n> of <m>) is required for striped datasets.
  end_thread   --thread 3--
END_THREADS
```

Writing Backup Scripts

A backup script consists of the following specific sections—global parameters, dataset descriptions, device specifications, dataset backup options, and thread descriptions. Each section is described below.

BACKUP_GLOBAL_PARAMETERS Section

The global parameters for a backup script are:

- volume_group
- expiration
- start_date
- start_time
- tape_mount_timeout

The volume_group and expiration parameters are required; start_date and start_time parameters are optional, but can be used to schedule a delayed backup. (See “Benefits of the Delayed Backup Feature” on page 114.) The tape_mount_timeout parameter default is no timeout. A timeout parameter may be useful if you run an unattended backup.

Volume_group

The volume_group global parameter is required. The format of the volume_group parameter is:

```
volume_group = <vg_identifier>;
```

where <vg_identifier> is a volume group identifier.

Note Do **not** enclose the identifier in double quotes.

Observe the following restrictions on the volume group identifier:

- Must be from 1 to 28 characters long
- May consist of any combination of ASCII uppercase letters, ASCII decimal digits, and the ASCII underscore character
- First character must be an ASCII uppercase letter

All tapes involved in a backup must have the same volume group name. You can ensure this by making the volume group name a global parameter.

Adopt a systematic convention for volume group names that indicates the most relevant information about the volume group.

For example, a volume group name might contain the following information:

- Name of the system backed up
- Day of week
- Morning or evening
- Cycle number

  If you back up on the recommended weekly cycle and keep four weeks of tapes, cycles will be numbered from 1 to 4.

- Online or offline backup
- Full or interval backup

Note A full backup must be performed before you perform the first interval backup. See **"Full versus Interval Backups" on page 107** for more information.

Using the information described above, a volume group name of BANDIT_MONPM3_ONLIVAL identifies an online interval backup, performed on Monday evening of cycle number 3 for the system named "bandit."

Expiration

The expiration global parameter is required.

The expiration global parameter specifies how long the backup tapes remain protected after the backup is made. Prior to the expiration date and time, EBR rejects any attempt to write another backup to unexpired tapes. The expiration parameter prevents tapes from being accidentally overwritten by EBR for a guaranteed minimum amount of time.

The format of the expiration global parameter is:

expiration = <integer> <units> ;

where <units> is one of the following:

- day
- days
- week
- weeks
- month (calculated as 31 days)

- months (each month calculated as 31 days)

For example, to protect your tapes from being overwritten for one month after the backup is created, specify the following:

```
expiration = 1 month;
```

Note If you specify expiration in months, EBR calculates the expiration date based on a 31-day month. For example, if your March 25, 2004, backup specifies expiration = 12 months, EBR calculates the expiration date as April 1, 2005.

If the expiration date crosses a daylight savings time period, EBR does not adjust for the one-hour difference in time. For example, in the twelve-month example above, if the backup start time is March 25, 2004, at 16:20:27, the expiration date and time are April 1, 2005, at 17:20:27.

Start_date

The start_date parameter is optional.

Use the start_date parameter to specify when you want a delayed backup to start. If you specify start_date, you must also specify the start_time parameter.

CAUTION Shutting down your system or recycling FileNet software results in a loss of delayed backup jobs waiting to run. Preferred methods for scheduling unattended, delayed backups are the UNIX crontab utility or the Windows Server AT command. See **"Alternate Delayed Backup Methods" on page 116** for more information.

The format of the start_date parameter is:

start_date = <year> / <month> / <day> ;

where, <year> is a four digit integer specifying the year, <month> is an integer from 1 to 12 specifying the month, and <day> is an integer from 1 to 31 specifying the day of the month. Do **not** enclose values in double quotes.

For example, to cause EBR to start the backup on December 13, 2004, specify the following:

start_date = 2004/12/13;

Even though the backup will not start until the specified start date and start time, the EBR process starts as soon as you run the script and opens a window for the user interface display. The window must remain open until the EBR operation scheduled to occur at the start date and time has **completed**.

EBR displays the following message for a delayed backup:

```
EBR: Job has been scheduled to run at Fri Dec 13
13:00:00 2004.
Please leave the window open for EBR user interface to
start.
```

Start_time

The start_time parameter is optional.

The start_time parameter specifies at what time the backup will start. If you specify the start_time parameter, you must also specify the start_date parameter.

The format of the start_time parameter is:

start_time = <hour> : <minute> ;

where <hour> is an integer specifying the starting hour in 24 hour time (noon = 12, 1 p.m. = 13, etc.), and <minute> is an integer specifying the starting minute of the hour. Do **not** enclose values in double quotes.

For example, to start the backup at 6:30 p.m., specify the following:

start_time = 18:30;

EBR waits until the time specified by start_time before starting the backup.

Tape_mount_timeout

The tape_mount_timeout parameter is optional. If you do not specify a tape_mount_timeout parameter, the default is no timeout and the tape mount operation does not abort.

Setting a tape_mount_timeout parameter allows you to define a period of time at the end of which the tape mount process will abort if the tape has not mounted successfully. The timeout feature is useful if you run an unattended backup and no user intervention is on hand to correct a tape error.

The tape mount process attempts to mount the tape until the tape_mount_timeout value expires. When the timeout value is reached and the tape does not successfully mount, the tape mount process aborts. Usually the problem is the wrong tape, a tape I/O error, or tape is not inserted.

The format of the tape_mount_timeout parameter is:

tape_mount_timeout = <integer_second>;

where <integer_second> is an integer specifying seconds. Do **not** enclose the value in double quotes. For example, to set a value of 300 seconds, specify the following:

tape_mount_timeout = 300;

On some platforms, such as AIX, you can configure your tape drive with the following options that direct the tape drive to retry open, read, and write operations until the specified time expires:

"Set timeout for the READ or WRITE command"

"Set delay after a FAILED command"

If a timeout value is also set, EBR does not abort the tape mount process until the time specified in the options above has expired. If the time specified in the tape drive configuration options is less than the tape_mount_timeout value, EBR first asks the user to insert the tape in the drive and then tries to open the tape. If the time specified in the tape drive configuration options is greater than or equal to the tape_mount_timeout value, EBR displays a tape offline error and aborts the tape mount operation.

DATASETS Section

In the DATASETS section of your backup script, you describe the datasets you want to back up. Dataset descriptions are in the form:

```
<dataset_name> : <dataset_type>
...
end_<dataset_type>
```

where <dataset_name> is an identifier and <dataset_type> is the type of dataset.

The dataset identifier is the name of the dataset assigned by the author of the script. The identifier must begin with a letter followed by any combination of ASCII upper and lowercase letters, ASCII decimal digits, or the ASCII underscore character, up to a maximum of 80 characters.

The following table lists the supported dataset types. Dataset types are **case sensitive** so be sure to enter them in your script exactly as they appear in the table:

Supported Dataset Types

Type	Description
partition	raw disk partition
MKF	MKF database
Oracle	Oracle database
cache	FileNet magnetic disk cache (for example, a page cache)

The sections below further describe each dataset type.

Note You can create and save a DATASETS section of your script in a separate ASCII file that all backup and restore scripts use. The script uses the include preprocessor directive to include the ASCII file. See **"Include File Descriptions" on page 397** for a sample include file.

Raw Disk Partitions

An example declaration of a raw disk partition dataset is the following:

```
y: partition
    location = “TapeServer1:bandit:FileNet”;
    filename = “/x/y”;
    start_block = 0;
    block_size = 1024 bytes;
    num_blocks = 100;
end_partition
```

In this example, the script’s author assigned the identifier **y** as the name of this dataset and assigned it a disk type of partition. An object named TapeServer1:bandit:FileNet resides in the network clearinghouse (NCH) database and one of its attributes is the network address of the host containing the disk partition “/x/y”. The raw partition consists of a number of 1024-byte blocks. The dataset is 100 blocks long, and starts at block 0.

Values for location and filename are required.

The start_block, block_size, and num_blocks values are optional. You can use start_block to back up specific blocks of a partition.

The legal values for block_size are 512, 1024 (the default), 2048, and 4096. The default for start_block is 0. The num_blocks parameter defaults to all blocks in the raw partition.

MKF Databases

For MKF databases, you specify only two parameters: location (so RPCs can be performed) and the MKF base_data_file name. For

example, if the script writer chooses the identifier **perm** as the name of the MKF permanent database, the database description is as follows:

```
perm: MKF
    location = “TapeServer1:bandit:FileNet”;
    base_data_file = “/fnsw/dev/1/permanent_db0”;
end_MKF
```

In the example above, the “TapeServer1:bandit:FileNet” NCH database object has the network address value of the host containing the raw partition /fnsw/dev/1/permanent_db0 (the MKF permanent database on that host).

An MKF transient database is very closely related to a particular magnetic disk cache and that cache cannot be rolled forward. Because of these special characteristics, the transient database requires some special handling in the backup script. The dataset definitions of the transient database and its related cache must reference each other. You do this by (1) specifically identifying an MKF database as a transient database and (2) specifying the script dataset name of the associated transient database when setting up the disk cache description.

You must use the transient_db keyword for a transient database as shown in the following example:

```
tran: MKF
    location = “TapeServer1:bandit:FileNet”;
    base_data_file = “/fnsw/dev/1/transient_db0”;
    transient_db;
end_MKF
```

Oracle Databases

EBR can only back up Oracle databases used by FileNet Image Services and eProcess software. For FileNet databases managed by Oracle, specify in the DATASETS section of your backup script the disk service name and the full path name of the directory that is to contain the signature file.

Oracle Signature File

EBR uses the signature file in performing Oracle interval backups. The file system that holds the signature file directory must have enough disk space to store about 3.2% of the size of the Oracle database data files (not including Oracle redo logs). We recommend that you do not store other files in the signature file directory.

An example of an Oracle index database description is the following:

```
inxdb: Oracle
    location = “TapeServer1:bandit:FileNet”;
    signature_file_directory = “/fnsw/local/tmp/ora_sig”;
end_Oracle
```

The dataset name assigned by the script author is inxdb. The location of the host providing tape service is TapeServer1:bandit:FileNet. The signature file is located in the /fnsw/local/tmp/ora_sig/ directory. Replace this example directory name with the name of the directory in which you store the signature file.

Note Turn on archive redo log mode for your Oracle databases. Specify that Oracle archive its redo logs to magnetic disk. For details on how to turn on archive redo log mode, see **“Appendix F – Enabling Archive Log Mode” on page 377** or refer to your Oracle documentation.

The Oracle SID (system identifier) is always **IDB** (for Index DataBase). The name of the signature file is **oraIDB_sig**.

During a full backup or full restore, **oraIDB_sig.new** is the name of the new signature file being created to replace the old one.

At the end of the operation, EBR renames the new signature file to the old name and deletes the old signature file. Together, the old and the new signature files require 3.2% of the size of the Oracle database data files.

Disk Cache

When setting up the description of a disk cache, you need to specify only the NCH name of the cache and the script dataset names of the transient database that corresponds with the named cache, the permanent database, and the security database.

Note The location can be specified in one of four formats: three-part NCH name, NCH domain name, TCP/IP address, or host name. In addition, EBR recognizes 3.3.1 release syntax (cache_name = “page_cache1:bandit:FileNet”) or the syntax used in 3.4.0 and later as shown below (location = “page_cache1:bandit:FileNet”).

The following is an example of a disk cache description:

```
cache1: cache
    location = “page_cache1:bandit:FileNet”;
    transient_db = tran;
    permanent_db = perm;
    security_db = sec;
end_cache
```

In the example, the name of the dataset is cache1 and the NCH name of the cache is page_cache1:bandit:FileNet. The script dataset name of the transient database controlling the cache is tran.

Maintaining Cache Data Integrity through Order Constraints

When you back up or restore a cache, EBR uses the order_constraints subsection of the DATASETS section to enforce that certain dependent datasets, such as the transient database and the cache, are backed up and restored in a certain order. In this way, EBR protects data and maintains dataset synchronization.

You must include the order_constraints subsection in the DATASETS section of your script when you are backing up cache. Only one order_constraints subsection is permitted in the DATASETS section, and order_constraints must always be the **last** subsection in the DATASETS section.

Tip When you use the EBR_genscript tool to build a datasets definition file, the order_constraints subsection is automatically built.

As an example, the order_constraints subsection for the cache description under **"Disk Cache" on page 193** is the following:

```
order_constraints
    sec, perm, tran before cache;
end_order_constraints
```

You can specify multiple cases of the **before** keyword in an order_constraints subsection. For example, the following order_constraints subsection accomplishes the same result as the example above:

```
order_constraints
    sec before cache;
    perm before cache;
    tran before cache;
end_order_constraints
```

The order of the dataset backup operation you specify in order_constraints determines the sequence in which the backups (or restores) occur and EBR enforces that sequence. Although the cache dataset must always be the **last** item in the list, no particular sequence is enforced for dataset names to the left of the **before** keyword.

For more information about constraints on backing up and restoring the disk cache and its transient database, see **"Constraints" on page 123**.

If you are restoring cache to a system configured with multiple storage library servers, see **"Cache Restore Constraints for Multiple Server Systems" on page 128**.

DEVICE_SPECIFICATION Section

In the DEVICE_SPECIFICATION section, you define one of three device types—tape or tape library or magnetic disk—as the destination for backup data. You then specify the device type to use when you define your threads.

The general form of the device specification for tape is:

```
TAPE_DEV1: tape_drive
    location = "<host_location>";
    drive_name_rewind = "<full_path_name>" ;
    drive_name_no_rewind = "<full_path_name>" ;
```

```
        drive_type = <drive_type>;
    end_tape_drive
```

where <host_location> is the name, enclosed in double quotes of the host on which the tape drive is located and <full_path_name> is the full path name, enclosed in double quotes, of either the special file for rewind-on-close behavior or of the do-not-rewind-on-close special file. Both are required.

The location statement is required and specifies the system providing tape service for the backup. For example, if you are backing up on System A using the tape service on System B, you could specify the following for location:

```
location = “TapeServer1:SystemB:FileNet”
```

You can specify the tape service name, enclosed in double quotes, in any of the legal formats (see **“Specifying Network Host Locations” on page 154**). The example above uses the three-part NCH name.

The general form of the device specification for tape library is:

```
LIB1: tape_library
    location = “<host_location>”;
    library_device = “<full_path_name>” ;
    library_type ={Exabyte | “EXB-210” | “EXB-218” | “EXB-220”}
    <tape_library_reservation>

    tape_drives
      num_drives = <num_drives>;
      drive <num_drives>
        drive_name_rewind = “<full_path_name>” ;
        drive_name_no_rewind = “<full_path_name>” ;
```

```
            drive_type = <drive_type>;
        end_tape_drives
    end_tape_library
```

The general form of the device specification for disk file is:

```
DISK_FILE1: disk_file
    location = “<host_location>”;
    filename = “<full_path_name>”;
end_disk_file
```

where <full_path_name> is the full path name, enclosed in double quotes, of the disk file to be used to store the backup.

Note The DEVICE_SPECIFICATION can also be kept in a separate ASCII file and referenced in the script via an include statement.

The table below lists examples of rewind and no-rewind tape device file names for each supported platform.

Tape Device File Names

Platform	Rewind File Name	No-Rewind File Name
AIX/6000	/dev/rmt0	/dev/rmt0.1
HP-UX	/dev/rmt/0m	/dev/rmt/0mn
Solaris	/dev/rmt/0mb (BSD) /dev/rmt/0m (non-BSD)	/dev/rmt/0mbn (BSD) /dev/rmt/0mn (non-BSD)
Windows Server	tape0	tape0

Substitute a keyword value for <drive_type>. Do **not** enclose the <drive_type> keyword value in double quotes. Keywords for the supported drive types are listed in the table below.

Tape Drive Type Keyword Values

Keyword Value	Description
8mm	8 millimeter cartridge tape
4mm	4 millimeter cartridge tape
DAT	Digital audio tape, a synonym for 4mm
QIC	Quarter-inch cartridge

Keywords for the <lib_device_type> are listed in the table below. The <lib_device_type> is specified only for labeling tape in a tape library. The keyword may be enclosed in double quotes, except the Exabyte keyword does not require double quotes.

Tape Library Type Keyword Values

Keyword Value	Description
“EXB-210”	Exabyte 8 millimeter cartridge tape library
“EXB-218”	Exabyte 4 millimeter cartridge tape library
“EXB-220”	Exabyte 8 millimeter “Mammoth” cartridge tape library
Exabyte	Refers to any of the above Exabyte tape libraries and can be used if you are not sure of the type

You must specify the same tape device type in both the EBR_label command and your backup script. Failing to do so may result in an inability to restore from the backup tape. See **“Selecting Tape Device Type” on page 132** for more information.

The following example describes a tape backup device in an AIX/6000 environment:

```
TAPE_DEV1: tape_drive
    location = “coyote”;
    drive_name_rewind = “/dev/rmt0”;
    drive_name_no_rewind = “/dev/rmt0.1”;
    drive_type = 8mm;
end_tape_drive
```

Then in the Threads section, the following describes how you specify tape_drive as the tape backup device illustrated above:

```
Thread 1
      device = tape_drive;
      volume_serial = TAPE_SER1;
      datasets
         perm (part 1 of 3);
 end_thread
```

In the Solaris environment, when specifying the rewind and no-rewind file names for tape devices, use the actual device name for the BSD or non-BSD device. Do **not** use the link name for the device. The following example specifies a valid file name for a BSD tape device:

```
TAPE_DEV1: tape_drive
    location = “coyote”;
    drive_name_rewind = “/dev/rmt/0mb”;
    drive_name_no_rewind = “/dev/rmt/0mbn”;
    drive_type = 8mm;
end_tape_drive
```

If you inadvertently specify a link name such as /dev/rst12 that points to the tape device name /dev/rmt/0mb, EBR issues an error message containing syntax guidelines. For more information on tape device names in the Solaris environment, see **“Specifying Tape Drive Device Names on Solaris Servers” on page 134**.

The following example describes a tape library backup device in an AIX/6000 environment:

```
LIB1: tape_library
  location = “coyote”;
    library_device = “/dev/chgr0”;
    library_type = “EXB-210”;

    tape_drives
      num_drives = 1;
      drive 1
        drive_name_rewind = “/dev/rmt0”;
     drive_name_no_rewind = “/dev/rmt0.1”;
     drive_type = 8mm;
      end_tape_drive
```

Then in the Threads section, the following describes how you specify a tape_library that has a bar code reader as the tape library backup device illustrated above:

```
Thread 1
      device = LIB1 (drive1);
      volume_serial = BAR_0002000;
      datasets
         part, sec, perm,
         tran, cache,
```

```
        inxdb (part 1 of 2);
end_thread
```

The following examples describe a disk file, prelabeled with EBR_ label, to receive the backup:

UNIX

For UNIX platforms:

```
DISK_FILE1: disk_file
    location = “test_domain:FileNet”;
    filename = “/tmp/mybackup”;
end_disk_file
```

WIN

For Windows Server platforms:

```
DISK_FILE1: disk_file
    location = “test_domain:FileNet”;
    filename = “<drive>=\tmp\mybackup”;
end_disk_file
```

EBR supports backing up to and restoring from disk files. However, you will probably use tape for production backups. Backups to disk files are most useful for testing and debugging your scripts.

BACKUP_OPTIONS Section

The backup options you specify depend upon the type of dataset you are backing up. The general form of backup options is:

```
<dataset_name> : backup_options
 ...
end_backup_options
```

where <dataset_name> is the name of the dataset in the DATASETS section.

The options possible for each type of dataset are described in the following topics.

Raw Disk Partitions

No backup options exist for raw disk partitions.

MKF Databases

For most MKF databases, you must choose either the online_backup or offline_backup option. You must also choose either the full_backup or interval_backup option. (The exception is the transient database, for which you can only choose the full_backup and offline_backup options.) These option pairs are independent of each other so four option combinations are possible.

Note You must perform a full backup before you can perform the first interval backup of a database.

For example, to perform a full, offline backup of the MKF permanent database (the dataset named perm), specify the following:

```
perm: backup_options
    full_backup;
    offline_backup;
end_backup_options
```

To perform an online interval backup of the permanent database, specify the following:

```
perm: backup_options
    interval_backup;
    online_backup;
end_backup_options
```

To perform a backup of the MKF transient database, specify **only** the following options:

```
tran: backup_options
    full_backup;
    offline_backup;
end_backup_options
```

Oracle Databases

For Oracle databases, you must choose either the online_backup or offline_backup option. You must also choose either the full_backup or interval_backup option.

Note You must perform a full backup before you can perform the first interval backup of a database.

Another required parameter, archive_redo_log_retention, tells EBR how long to retain an archive redo log on magnetic disk. The minimum number of days you can specify is two. The following parameter specifies a retention period of four days.

```
archive_redo_log_retention = 4 days;
```

EBR searches magnetic disk for old archived redo log files in the archive redo log directory you specified during Oracle setup (in the log_archive_dest parameter of Oracle's initialization parameter file, init.ora). At the end of the job, the program deletes any files older than

the specified number of days. This automatic deletion prevents the magnetic disk file system from filling up.

Tip If you choose to retain older archive redo logs for disaster recovery purposes, you must copy them to tape manually. EBR does not back up the archive redo logs located in the directory you specified in the log_archive_dest parameter because a successful full backup and regularly scheduled interval backups pick up the changes recorded in the archive logs. Older archive redo logs are only useful in the highly unlikely event that a triple failure occurs: the disk containing your Oracle database crashes, your last full backup tape is lost or destroyed, and the disk containing your Oracle archive redo log crashes.

This triple failure scenario can be prevented if you place your archive redo log on a different spindle than your database. However, if you place them on the same spindle, consider manually backing up older archive redo logs to tape.

The following example script performs a full online backup of the Oracle dataset **inxdb** and deletes from magnetic disk any archive redo logs older than 4 days:

```
inxdb: backup_options
    full_backup;
    online_backup;
    archive_redo_log_retention = 4 days;
end_backup_options
```

Disk Cache

Select the full_backup option. No other backup options are needed.

The following example script backs up all the locked objects in the cache1 dataset:

```
cache1: backup_options
full_backup;
end_backup_options
```

For details about constraints on backing up and restoring a disk cache and its associated transient database, see **"Constraints" on page 123**.

THREADS Section

Each tape cartridge used in a backup is associated with one thread. Describe the threads in the THREADS section of your script. The general form of the THREADS section is:

```
THREADS
    num_threads = <integer> ;
    thread 1
    ...
    end_thread
    thread 2
    ...
    end_thread
    ...
END_THREADS
```

where <integer> is the number of threads specified as an integer. The number of threads is followed by a specification for each thread. The individual threads are assigned consecutive integers starting at 1 and thread specifications must occur in ascending thread number order.

Note Running multiple threads requires more main memory than running a single thread. Insufficient memory causes EBR process failures, program termination, and requires you to recycle FileNet software. To ensure that sufficient memory is available for the number of threads you specify in your script, see **"Appendix G – Memory Requirements" on page 384**.

The general form of the specification of each thread is:

```
thread <integer>
    device = tape_drive or tape_library or disk_file;
    volume_serial = <tape_serial_number> ;
    datasets <list_of_datasets> ;
end_thread
```

where <integer> is the thread number.

When backing up cache, be aware of the effect of order_constraints on the thread specification. Incorrect specification of the threads for cache and transient database backup can result in syntax errors.

For example, if you back up cache and its associated transient database using two threads, assign thread 1 to the transient database and thread 2 to cache, as shown in the partial description below:

```
thread 1
    device = TAPE_DEV1;
    volume_serial = VG1;
    datasets tran;
end_thread

thread 2
    device = TAPE_DEV2 ;
```

```
    volume_serial = VG2 ;
    datasets cache;
end_thread
```

The backup completes in the proper order, enforced by EBR, to maintain dataset synchronization between the cache and its associated transient database.

For comparison, examine the thread specification of the following two **incorrect** examples:

Incorrect Specification of Order Constraints - Example 1

```
thread 1
    device = TAPE_DEV1;
    volume_serial = VG1;
    datasets cache;
end_thread

thread 2
    device = TAPE_DEV2 ;
    volume_serial = VG2 ;
    datasets tran;
end_thread
```

Incorrect Specification of Order Constraints - Example 2

```
thread 1
    device = TAPE_DEV1;
    volume_serial = VG1;
    datasets perm, sec, cache;
end_thread
```

```
thread 2
    device = TAPE_DEV2;
    volume_serial = VG2 ;
    datasets tran;
end_thread
```

Both Example 1 and Example 2 will cause EBR to generate the following syntax error message:

```
Ordering constraint violated across thread 1 and 2 for
datasets cache and tran. If tape drive were shared,
deadlock would result.
```

For more information on order constraints when backing up cache, see **"Order Constraints" on page 123**.

Device Type

You must specify a device type as the destination for backup data for each of your threads. The device types, tape or tape library or magnetic disk, were already defined in the DEVICE_SPECIFICATION section.

You must specify the same tape device type in both the EBR_label command and your backup script. Failing to do so may result in an inability to restore from the backup tape. See **"Selecting Tape Device Type" on page 132** for more information.

The following example describes how you specify tape_drive as the tape backup device:

```
Thread 1
      device = tape_drive;
```

```
        volume_serial = TAPE_SER1;
        datasets
           perm (part 1 of 3);
 end_thread
```

The next example describes how you specify a tape_library that has a bar code reader as the tape library backup device:

```
 Thread 1
        device = LIB1 (drive1);
        volume_serial = BAR_0002000;
        datasets
           part, sec, perm,
           tran, cache,
           inxdb (part 1 of 2);
 end_thread
```

Volume Serial Number

With the volume_serial option, you specify the tape serial number of the tape to which the backup data is to be written.

Every tape should have a tape serial number that is distinct from all other tapes in your enterprise. EBR enforces the rule that all tapes involved in a backup have tape serial numbers that are unique.

Tape serial numbers are identifiers, but with the following additional restrictions:

- A tape serial number consists of one to 12 characters.
- Characters of the tape serial number must consist of any combination of ASCII uppercase letters, ASCII decimal digits, or the ASCII underscore character. (Do not use lowercase letters.)

- The first character of the tape serial number must be an ASCII uppercase letter.

For a tape library labeling, if a tape has a bar code label and a bar code reader is present, the tape serial number must be entered as:

BAR_ascii_characters

For example, BAR_0002000 or BAR_CNL001, where the ascii_characters part of the serial number must be the same as the bar code label.

We suggest that you systematically assign tape serial numbers. The tape serial number could include an abbreviated version of the system name followed by the year, month, and day, and a sequence number for that day. Once you have given the systems in your enterprise agreed-upon abbreviations, you can decentralize the creation of new tape serial numbers.

As an example, the abbreviation for the system called “bandit” is BDT. The first tape created on bandit on December 8, 2004 would be tape serial number BDT041208001.

We recommend both the tape serial number and volume group name be written on a human-readable external label affixed to the tape cartridge.

Tapes are assigned tape serial numbers and volume group names by the EBR_label program. For information on the EBR_label program, see **“Appendix A – Programs and Utilities” on page 265**.

Datasets

The datasets description of the THREADS section lists the datasets that the corresponding thread is to back up. The general form of the datasets description is:

```
datasets <list_of_datasets>
```

where <list_of_datasets> is a comma-separated list of dataset names from the DATASETS section, for example:

```
datasets y, tran, inxdb;
```

Since blanks and end-of-line markers between tokens are not significant, you can also write the datasets information as follows:

```
datasets
    y,
    tran,
    inxdb;
```

When a dataset list involves a backup of cache and one or more of the MKF permanent, security, and transient databases, cache must **always** be the last item in the list, as shown in the example below:

```
datasets sec, perm, inxdb, tran, cache;
```

Special Considerations for Listing Striped Datasets

If your large datasets have been striped (broken into smaller parts), you must take into consideration special requirements for the dataset list of the THREADS section. When using striped datasets, the part number must appear after the dataset name.

For example, suppose you use two threads and the inxdb dataset is broken into two parts. Your script would look similar to the following (some portions of the individual thread descriptions have been omitted in the example):

```
THREADS
   num_threads = 2;
   thread 1
        ...
      datasets inxdb (part 1 of 2);
   end_thread
   thread 2
      ...
      datasets inxdb (part 2 of 2);
   end_thread
END_THREADS
```

In this example, the first part of inxdb is assigned to thread 1 and the second part is assigned to thread 2. The syntax

```
(part <n> of <m>)
```

describes the part number (<n>) and total number of parts (<m>).

The parts, like threads, are numbered from 1. As the part number increases, the thread number of the part must also increase.

The following script is an example of an **illegal** specification:

```
THREADS
   num_threads = 2;
   thread 1
        ...
```

```
        datasets inxdb (part 2 of 2);
    end_thread
    thread 2
        ...
        datasets inxdb (part 1 of 2);
    end_thread
END_THREADS
```

The preceding example would generate errors because, as the part number increases from 1 to 2, the thread number decreases from 2 to 1. Such a decrease violates the ordering restriction, which is enforced when the script is parsed.

EBR is coded so that the first part (part 1) performs any necessary dataset initialization and the last (the highest numbered part) performs the final termination code. For example, for databases, the last part rolls the database forward to the last transaction. All parts greater than the first part wait for the first part to finish the initialization before they start. The last part waits for all other parts before it performs the termination code. This ordering restriction prevents deadlock and makes it possible for all parts to share a single tape drive.

The number of dataset parts on a tape is limited to one. If you attempt to assign more than one part of a dataset to a single tape, a script syntax error results.

Note If EBR fails with a core dump due to striping errors, save the core dump file and any related error messages. Then call your service representative for troubleshooting assistance.

Writing Restore Scripts

In this section, we describe each of the sections of a restore script: global parameters, dataset descriptions, device specification, dataset restore options, and thread descriptions.

A restore script is very similar to a backup script. To save time and coding effort, make a copy of your backup script and edit the appropriate sections to reflect restore options. Save the edited script as a restore script. See **"Tips for Creating Scripts" on page 233** for detailed procedures and a list of necessary changes.

You can also use and modify the FileNet-provided sample scripts in /fnsw/local/EBR/samples on UNIX platforms or, for the Windows Server platform, in \fnsw_loc\EBR\samples.

RESTORE_GLOBAL PARAMETERS Section

The volume group name is required and is the only global parameter you need to specify for a restore. The syntax of the volume group name is:

```
volume_group = <volume_group_name> ;
```

For example, the volume group name below describes the online interval backups performed on Monday at 3 p.m. of cycle 3 for the system named bandit:

```
volume_group = BANDIT_MONPM3_ONLIVAL ;
```

DATASETS Section

The DATASETS section is exactly the same for restore scripts as for backup scripts. We recommend that you create and save a DATASETS

section of your script in a separate ASCII file and that all backup and restore scripts use the include preprocessor directive to include this file.

For details, see the **"DATASETS Section" on page 188**.

DEVICE_SPECIFICATION Section

The DEVICE SPECIFICATION section is exactly the same for restore scripts as for backup scripts. We recommend that you create and save a DEVICE SPECIFICATION section of your script in a separate ASCII file and that all backup and restore scripts use the include preprocessor directive to include this file.

For details, see the **"DEVICE_SPECIFICATION Section" on page 195**.

RESTORE_OPTIONS Section

The restore options you specify depend on the type of dataset you are restoring. The general form of the RESTORE_OPTIONS section is:

```
RESTORE_OPTIONS
   <dataset_name> : restore_options
   ...
   end_restore_options
...
END_RESTORE_OPTIONS
```

where <dataset_name> is the name of the dataset in the DATASETS section of your script.

The options for each type of dataset are described in the following topics.

Raw Disk Partitions

You can optionally restore a raw disk partition onto a partition with a different name either on the same host or a different host. The syntax is:

```
<dataset_name>: restore_options
   restore_onto “<full_path_name>” ;
end_restore_options
```

where <full_path_name> is the full path name, enclosed in double quotes, of the disk partition that will receive the data. If the partition is being restored to a partition on a different host, specify the receiving host in the DATASETS section of the script. Using the restore_onto option, you can move partitions to other systems quickly and easily.

The full path name for the restore_onto option is required. If you do not enter the full path name, EBR generates the following syntax error:

‘filename = “<full_path_name>” ;’ expected

Volume control block information within a partition contains the control information for a partition. The restore_onto option for restoring a partition causes both control block information and data to be copied from the original partition to the new partition. In contrast, the reconfigure_onto option for MKF databases copies only the data (not control block information) from the original partition to the new partition. (See **“MKF Databases” on page 218** for details of the reconfigure_onto option.)

Tip AIX/6000 uses the first 4K bytes of the partition as the volume control block. If you use the restore_onto option, this volume control block information is overwritten. If you do not want to overwrite the volume control block information during restore of an AIX/6000 raw partition,

specify the start_block and block_size during backup of the raw partition. The product of start_block and block_size must be 4096 bytes.

To restore a raw partition dataset **y** onto a different partition named **/x/z**, specify the following in the RESTORE_OPTIONS section of your script:

```
y: restore_options
    restore_onto “/x/z”;
end_restore_options
```

Note If the partition you are restoring to does not exist, EBR creates a regular file (not a partition) and restores the data to the new file.

To restore a raw partition dataset **y** from the system called bandit onto a new partition named **/x/z** on the system called rojo, specify the following in the RESTORE_OPTIONS section of your script:

```
y: restore_options
    restore_onto “/x/z”;
end_restore_options
```

Then specify the receiving host in the datasets section of your restore script, as follows:

```
y: partition
     location = “TapeServer1:rojo:FileNet”;
     filename = “/x/z”;
end_partition
```

MKF Databases

If you follow the recommended backup discipline (weekly full backups and daily interval backups), the restore process consists of restoring the most recent full backup followed by restoring the most recent interval backup.

Note EBR does not support interval backups of the transient database and its associated cache. Restore only full backups of the transient database. The interval restore information in this section applies only to the permanent and security MKF databases.

In your script, you must specify whether you are restoring a full backup or an interval backup. If you are restoring a full backup, you may or may not be restoring an interval backup next. With the exception of the transient database and its associated cache, for which EBR performs only restores of full backups, the usual case is that an interval restore follows the restore of a full backup. Restore the most recent full backup tape followed by the **most recent** interval backup tape. If you don't have an interval backup tape, restore only the full backup tape.

You must specify in your restore script which of these two cases pertains (an interval restore does or does not follow the restore of the full backup) so EBR knows whether to roll the database forward to the last transaction after the full restore or to wait until after the interval backup is restored to perform the rollforward operation.

The interval restore statement must appear immediately after the full restore statement as shown in the example below:

```
RESTORE_OPTIONS
   permdb : restore_options
      full_restore;
```

```
        interval_restore_follows = false;
        reconfigure_onto “/tmp/test”;
    end_restore_options

  inxdb : restore_options
      full_restore;
      interval_restore_follows = false;
      restore_control_file = true;
      restore_redo_logs = true;
  end_restore_options
...
END_RESTORE_OPTIONS
```

If the interval restore statement does not appear immediately after the full restore statement, EBR issues a syntax error:

> SYNTAX ERROR on line xx, column x, of Backup/Restore script file “<filename>”:
> “interval_restore_follows = true;” or “interval_restore_follows = false;” expected.

To remind you to perform an interval restore if necessary, EBR issues the following message at the end of the restore of a full backup:

> Full restore has been completed. Restore interval.

The message is posted to the user interface display and to the progress log.

CAUTION If you specify that no interval restore follows the restore of the full backup, yet you actually need to do an interval restore, the rollforward attempt will probably fail because the recovery log did not splice in at the correct time. At that point, a rollforward of the database to the last

transaction is no longer possible. Processing is lost and the database is unsynchronized with the other databases.

This situation does not occur for Oracle databases. However, if you inadvertently cause this situation for an MKF database, call your service representative immediately.

The syntax of MKF dataset restore options varies depending on one of the following two cases, one of which must be specified in your script:

Case 1: You are restoring a full backup with no interval backup:

```
<dataset_name> : restore_options
    full_restore;
    interval_restore_follows = false;
end_restore_options
```

After the full restore completes, EBR rolls the database forward to the last transaction, if possible.

Case 2: You are restoring a full backup and an interval backup:

```
<dataset_name> : restore_options
    full_restore;
    interval_restore_follows = true;
end_restore_options
```

Since interval_restore_follows is set to true, an interval restore **must** be the next script you run. The syntax for the interval restore is the following:

```
<dataset_name> : restore_options
    interval_restore;
end_restore_options
```

When the interval restore completes, EBR automatically rolls the database forward to the last transaction, if possible.

CAUTION EBR makes no attempt to roll the database forward to the last transaction **until** the restore of the interval backup completes.

The interval restore must occur **immediately** after the completion of the full restore of the dataset. If the MKF database is opened normally before the interval restore is performed, EBR will be unable to roll the database forward to the last transaction. Restarting the FileNet software opens the MKF databases, so do not restart FileNet software until the interval restore completes.

If a restore of the MKF permanent database successfully rolled the database forward to the last complete transaction, you are finished with the restore. However, if the permanent database was not successfully rolled forward, you must run the SNT_update utility to ensure that duplicate document numbers will not be assigned when document entry resumes. For more information about SNT_update, see **"Scalar Numbers Table Update After Restore Failure" on page 259**.

The reconfigure_onto Option

An optional restore parameter for the MKF permanent and security databases, reconfigure_onto, allows you to restore an MKF database on top of a different MKF database that has the same logical description for the data, but may differ in file names, file sizes, number of files, existence of files, whether files or partitions are used, and certain data-

base global parameters. Using this option, you can move the MKF permanent or security database to other systems quickly and easily and make changes to the files or partitions of the database. You **cannot** use the reconfigure_onto option for the transient database.

If the database you are restoring is larger than the database being restored onto, the restore may fail depending on the amount of in-use data that was backed up. The number of bytes of in-use data that was backed up must fit into the receiving database. EBR does not attempt to restore unused areas of the database to the receiving database. If the amount of in-use data exceeds the size of the receiving database, the restore fails.

Using the reconfigure_onto Option

You must perform the following steps to use the reconfigure_onto option.

1 Create an MKF DDL description for the new database.

From fn_edit in UNIX platforms or the FileNet Configuration Editor in Windows Server platforms, select the Datasets tab and the MKF Databases tab to view the description of the new database.

In the DDL for the new database, you must use the same logical description (the same TABLES section) as was used for the old database, but a different FILES section. The DDL for the MKF database is in /fnsw/local/sd/1directory for UNIX platforms or \fnsw_loc\sd\1 directory for Windows Server platforms. For example, the DDL for the MKF permanent database in a Windows Server platform is \fnsw_loc\sd\1\permanent.ddl.

The following example shows a DDL after the FILES section is modified:

```
PARAMETERS

(
number_of_buffers = 256,
max_concurrent_transactions = 3,
max_concurrent_long_transactions = 1,
read_after_write = set,
max_record_types = 32,
max_items_per_record = 32,
overwrite_rl_action = warning_message,
rl_update_frequency = 230
);

FILES
(
target_station "DocServer";
base data     partition "/tmp/new_database" (blocks = 102400);
recovery_log  partition "/tmp/new_database_rl0" (start = 0, blocks =
40960);

);
#include "/fnsw/lib/perm_db.desc"
```

2 Build the new DDL file with the following command:

fn_build –f <ddl_file>

where <ddl_file> is the name of the new DDL file you created.

3 Initialize the new database with the following command:

fn_util <init_option>

where <init_option> is the initialization option for the MKF dataset you are initializing:

initperm for the permanent database
initsec for the security database

4 Perform a restore using the reconfigure_onto option.

The syntax for the option is:

reconfigure_onto “<base_data_file_name>” ;

where <base_data_file_name> is the full path name, enclosed in double quotes, of the base data file for the new database.

The full path name for the reconfigure_onto option is required. If you do not enter the full path name, EBR generates the following syntax error:

‘base_data_file = “<full_path_name>” ;’ expected

Full path names must follow the format style of the examples below:

On Windows Server: reconfigure_onto “<drive>: \tmp\new_database”;

On UNIX platforms: reconfigure_onto “/tmp/new_database”;

Note When using the reconfigure_onto option, you must restore a full backup with no interval backup following. In other words, you must have **all** of the data you intend to restore contained in **one full backup**.

Examples of reconfigure_onto

The following example restores a backup of the MKF dataset **perm** onto a new MKF database with a base data file name of /tmp/new_database:

```
perm: restore_options
    full_restore;
    interval_restore_follows = false;
    reconfigure_onto “/tmp/new_database”;
end_restore_options
```

The new database must have previously been created and initialized with the same TABLES section as perm.

To specify that a database be restored to another system, you must specify the other system in the DATASETS section of the restore script. For example, to restore the perm database from the system called bandit to the system called rojo, specify the receiving host in the DATASETS section of your restore script, as follows:

```
perm: MKF
     location = “TapeServer1:rojo:FileNet”;
     base_data_file = “/fnsw/dev/1/permanent_db0”;
end_MKF
```

Oracle Databases

If you follow the recommended backup discipline (weekly full backups and daily interval backups), the restore process consists of restoring the last full backup followed by restoring the last interval backup.

In your script, you must specify whether you are restoring a full backup or an interval backup. If you are restoring a full backup, you may or

may not be restoring an interval backup next. The usual case is that a restore of the most recent interval backup tape follows the restore of a full backup tape. Restore the most recent full backup tape followed by the **most recent** interval backup tape. If you don't have an interval backup tape, restore only the full backup tape.

You must specify in your restore script which of these two cases pertains (either an interval restore does or does not follow the restore of the full backup) so EBR knows whether to roll the database forward to the last transaction after the full restore or to wait until after the interval backup is restored to perform the rollforward operation.

To remind you to perform an interval restore if necessary, EBR issues the following message at the end of the restore of a full backup:

> Full restore has been completed. Restore interval.

The message is posted to the user interface display and to the progress log.

The syntax of Oracle dataset restore options varies depending on one of the following two cases, one of which must be specified in your script:

Case 1: You are restoring a full backup with no interval backup:

```
<dataset_name> : restore_options
    full_restore;
    interval_restore_follows = false;
end_restore_options
```

When the full restore completes, EBR rolls the database forward to the last transaction, if possible.

Case 2: You are restoring a full backup and an interval backup:

```
<dataset_name> : restore_options
    full_restore;
    interval_restore_follows = true;
end_restore_options
```

EBR makes no attempt to roll the database forward to the last transaction until the restore of the interval backup completes.

Since interval_restore_follows is set to true, an interval restore **must** be the next script you run. The syntax of the interval restore is the following:

```
<dataset_name> : restore_options
    interval_restore;
end_restore_options
```

After the interval restore completes, EBR automatically rolls the database forward to the last transaction, if possible.

CAUTION The interval restore must occur **immediately** after the full restore of the dataset completes. If the Oracle database is opened normally before the interval backup is performed, EBR will be unable to roll the database forward to the last transaction. Restarting the FileNet software opens the index (Oracle) database, so do not restart FileNet software until the interval restore completes.

Help for Unusual Circumstances

Three options are available to help you deal with very unusual circumstances that may occur when restoring an Oracle database:

- The restore_control_file option restores the Oracle control file from tape if the disk copy of the control file is corrupted or lost.
- The restore_redo_logs option restores online redo logs from tape if the disk copy of the redo logs is corrupted or lost.
- The rollforward option controls the automatic or manual rollforward of the database (automatic rollforward is the default).

For an overview of the options available to you for unusual circumstances, see **"Appendix J – Restoring Oracle" on page 411**. Do not include these options in a restore script unless you encounter the specific problems described in the topics below.

CAUTION Use the restore_control_file, restore_redo_logs, and rollforward options only for **very unusual** restore situations. Misuse of these options can cause loss of data and loss of rollforward capability.

restore_control_file Option

When set to true, the restore_control_file option restores the Oracle control file from tape. Do **not** restore the control file from tape unless absolutely necessary.

Tip To avoid the need to restore control files from tape, duplicate the Oracle control file on a separate magnetic disk spindle. If one copy is bad, you can recover the control file by using a file copy utility to copy the good version over the bad one. Restore the control file from tape **only** if both copies are bad.

If only the control files are bad but data files and redo logs are intact, use Oracle commands to recreate the control file. See your *Oracle7*

Server Administrator's Guide or Oracle8 Server Backup and Recovery Guide for procedures.

As a result of restoring the control file, EBR will be unable to automatically rollforward the database. Your RDBMS Database Administrator must manually perform the rollforward of the database.

In addition, restoring the Oracle control files from tape using the restore_control_file option resets the numbering of archived redo logs to one. Because of this, you must save a copy then manually remove any magnetic disk-resident archived redo logs before you restore the Oracle control files from tape.

CAUTION Before taking any action that may open the Oracle database (for example, restarting FileNet software), copy all magnetic disk-resident archived redo logs to tape or to another magnetic disk-resident directory, then delete them from the original directory.

If you fail to remove **all** magnetic disk-resident archived redo logs before starting the Oracle database, Oracle rollforward leaves your database in an **unusable** state until the problem is corrected.

The syntax for the restore_control_file option is:

```
restore_control_file = { true | false } ;
```

The default is false. If restore_control_file is true, restore_redo_logs can be set to true or false. See "restore_redo_logs Option" below.

Note You must specify the restore_control_file option in the last restore script you run. For example, if you plan to run a full restore only, specify the restore_control_file option in that script. However, If you plan to run a full restore followed by an interval restore, specify the option in the

interval restore script. If you specify the restore_control_file option in the full restore script when you should have specified it in the interval restore script, EBR issues an error message stating that the control file on tape is older than the control file on disk.

restore_redo_logs Option

Oracle requires valid online redo logs to function. You should always try to use the magnetic disk-resident online redo logs to roll the Oracle database forward to the last transaction. If you restore the online redo logs from tape, you overwrite the disk copy of those logs and eliminate any chance of rolling the database forward to the last transaction. However, if the magnetic disk-resident copy of the redo logs is corrupted, you can set the restore_redo_logs option to true to restore the online redo logs from tape and allow Oracle processing to continue.

Note To restore online redo logs, you must use an **offline backup tape**. Online backups do not back up Oracle online redo logs.

The syntax for the restore_redo_logs option is:

```
restore_redo_logs = { true | false } ;
```

The default is false. If restore_redo_logs is set to true, you must also set restore_control_file to true. See **"restore_control_file Option" on page 228**.

Oracle Rollforward Option

The Oracle rollforward restore option controls automatic rollforward of the Oracle index database. Normally, rollforward occurs automatically (the default). Occasionally, an Oracle database rollforward operation

may terminate due to failures that cannot be resolved by EBR. If this occurs, you can set the Oracle rollforward option to false to restore the database without automatically rolling the database forward to the last transaction. You can then correct the problem and rerun the restore with rollforward = false. When the restore completes, manually recover (rollforward) the database using Oracle techniques described in the *Oracle7 Server Administrator's Guide or Oracle8 Server Backup and Recovery Guide*.

The syntax of the Oracle rollforward restore option is the following:

```
rollforward = {true | false};
```

The default setting is true.

If you are restoring an online backup and interval_restore_follows = true or rollforward = false, EBR always recovers the database to the point in time at which the online backup ended.

Note The Oracle restore options, restore_control_file and restore_redo_logs, are **independent** of the rollforward restore option. You can restore control files and redo logs and choose not to automatically rollforward the database. For example, you might do this because you want to manually rollforward the database or because you have encountered a problem for which EBR is unable to automatically rollforward the database. All of the following examples of the options are valid:

```
restore_control_file = true;
rollforward = false;

restore_control_file = true;
rollforward = true;
```

```
restore_control_file = true;
restore_redo_logs = true;
rollforward = false;
```

Disk Cache

You must restore the transient database **in the same script** as its associated cache.

Note EBR supports only full offline restore for cache. Interval restore is not currently supported for cache. Always specify interval_restore_follows = false.

The syntax for restoring a cache is:

```
cache1 = cache
    full_restore;
    interval_restore_follows = { true | false } ;
end_cache;
```

See **"Constraints" on page 123** for details on the constraints associated with restores of a disk cache and its transient database.

THREADS Section

The THREADS section for restore is exactly the same as the THREADS section for backup. However, you must use a text editor to delete the datasets you do **not** want to restore from the dataset list at the end of each thread definition.

For details, see the backup script **"THREADS Section" on page 205**.

Tips for Creating Scripts

In this section, we provide you with tips to help you create and test your backup and restore scripts.

- You can create your scripts using a text editor as described below or you can use the FileNet-supplied sample script files, modifying them for your own environment. Sample files are described in **"Appendix I – Sample Scripts and Backus-Naur Form Scripts" on page 394**.

- Use a text editor or the EBR_genscript utility to create files that describe your DATASETS and DEVICE_SPECIFICATIONS sections. Then use the **include** preprocessor directive in your backup and restore scripts to reference these files.

  If you use EBR_genscript to create these files, the result will be a dataset definition file (with a .ddf extension) and a device definition file (with a .dev extension).

- The THREADS section is exactly the same for backup and restore. Use a text editor if you need to modify it, for example, to delete datasets you do not want to restore.

- Write all your backup scripts. If you use parameters to specify the tape serial number and the volume group name, the same script can be used multiple times during the backup cycle.

- If you are backing up in a dual server environment, you can create one script that defines the datasets, backup options, and thread options for both servers. Then you can run that script on either server.

Creating Full Restore Scripts from Backup Scripts

You can develop your restore scripts from corresponding backup scripts. Use the following procedure to create full restore scripts.

1. Make a copy of the backup script.
2. Edit the copy to reflect your restore requirements.

   If you used the **include** preprocessor directive for the DATASETS or the DEVICE_SPECIFICATIONS section, you do not have to modify that section. However, if you do not want to restore all datasets contained in the include file, edit the THREADS section of your script to delete those datasets you do not want to restore.
3. Modify the global parameters section.

   Change the beginning and ending lines to:

   ```
   RESTORE_GLOBAL_PARAMETERS
   .
   .
   .
   END_RESTORE_GLOBAL_PARAMETERS
   ```
4. Delete all the backup global options **except** the volume group name.

   Volume group name is required as a restore global option.
5. Replace the dataset specific backup options section with a dataset specific restore options section.

   a. Change all occurrences of BACKUP_OPTIONS to RESTORE_OPTIONS.

b Change all occurrences of backup_options to restore_options.

c Change all occurrences of END_BACKUP_OPTIONS to END_RESTORE_OPTIONS.

d For databases, specify full_restore. If you have an interval backup tape, specify “interval_restore_follows = true”. Otherwise, set the option to false (“interval_restore_follows = false”).

Creating Interval Restore Scripts

From the restore script developed in **“Creating Full Restore Scripts from Backup Scripts” on page 234**, you can create interval restore scripts. Use the following procedure.

1. Make a copy of your full restore script.
2. In the RESTORE_OPTIONS section, change “full_restore;” to “interval_restore;” and delete the “interval_restore_follows = true” or “interval_restore_follows = false” statement.
3. Delete from the dataset list of the THREADS section those datasets you do not want to restore.
4. Save the edited restore script.

   When you run this script, EBR ignores all the extra dataset definitions and dataset-specific restore options.

Testing Your Script Syntax

Use the **–syntax** parameter of the EBR command to test the syntax of your script. EBR examines the parameters and options you specified in

your script for correct syntax without actually running the script. If EBR finds syntax errors, messages display to help you locate the error.

To use this option, enter the following command:

EBR –syntax @<script>

where <script> is the name of the backup or restore script you want to test.

Note If EBR reports syntax errors, the line number in the error message is usually either the line number containing the error or the line parsed after the line in error.

6
Running The Backup Script

EBR performs backups and restores according to the type of script passed to the program. In this chapter, we cover the following topics:

- The syntax of the command to start EBR
- Information you should know to prepare your system for a backup and to monitor the progress of the backup operation
- Procedures to perform a backup or cancel a backup

Backup Program Command Syntax

The command line parameters of the EBR program are:

EBR [–syntax] @<script> [<param> ...] [@<script> [<param> ...]] ...

The **–syntax** parameter permits you to check the syntax of your script before actually running the script. See **"Testing Your Script Syntax" on page 235** for more information.

Precede the backup script name with the @ sign with no intervening spaces. The script file is an ASCII file that you create as described in **Chapter 5, "Developing Your Scripts," on page 147**.

The script may contain parameters. Parameters in the script always start with a dollar sign ($). However, you must omit the $ when entering parameters on the command line.The value of all parameters must be

specified on the command line following the script name. The syntax is:

<parameter_name>=<value>

Embedded spaces are not allowed when entering parameters on the command line.

For UNIX environments, enter the command to start the backup on the command line of an open X window. The following example includes command line parameters enclosed in single and double quotes:

```
EBR  @test_back  'location="costa10" '
```

For Windows Server environments, enter the command to start the backup on the command line of a DOS shell window. For parameters that need to be within double quotes, you must insert a backslash character (\) before each double quote as shown in the following example:

```
EBR  @test.bk  location=\"iqant6\"
```

You can run a series of scripts sequentially by specifying another script name followed by its parameters. If any errors are encountered while running any script, EBR terminates and does not run the remaining scripts.

Preparing for Backup

The information in this section provides general information with which you should be familiar before you start a backup. Details of each point are in earlier chapters of this manual.

If you perform offline backups, use the FileNet Task Manager Stop button to shut down the FileNet software on the systems involved, followed by the Backup button to start the systems in backup mode.

Note For every attempt to run backup, you must select the Task Manager Stop button followed by the Task Manager Backup button. For example, if your first backup attempt fails and you try another backup, you must again select the Task Manager Stop and Backup buttons.

If you perform an offline backup in a dual server or multiserver environment, the servers must be shut down and restarted in a particular order as described in **"Dual Server Backup Procedure" on page 243** or **"Multiserver Backup Procedure" on page 246**.

If you perform online backups, the FileNet software must be running on the affected systems in normal production mode. In addition, if you perform **online backups of Oracle databases**, you must have previously enabled archive log mode. For both offline and online backup of Oracle databases, an Oracle signature file directory must exist.

CAUTION If you have modified the MKF database configuration by adding or removing MKF partitions or MKF recovery logs, do **not** perform an **interval** backup of the MKF databases (Transient, Permanent, and Security databases) until you have performed a full backup first. Otherwise, you will not be able to restore the interval backup.

You may want to run your backup in unattended mode, especially if the backup is a delayed job. See **"Unattended Backups" on page 120** for more information.

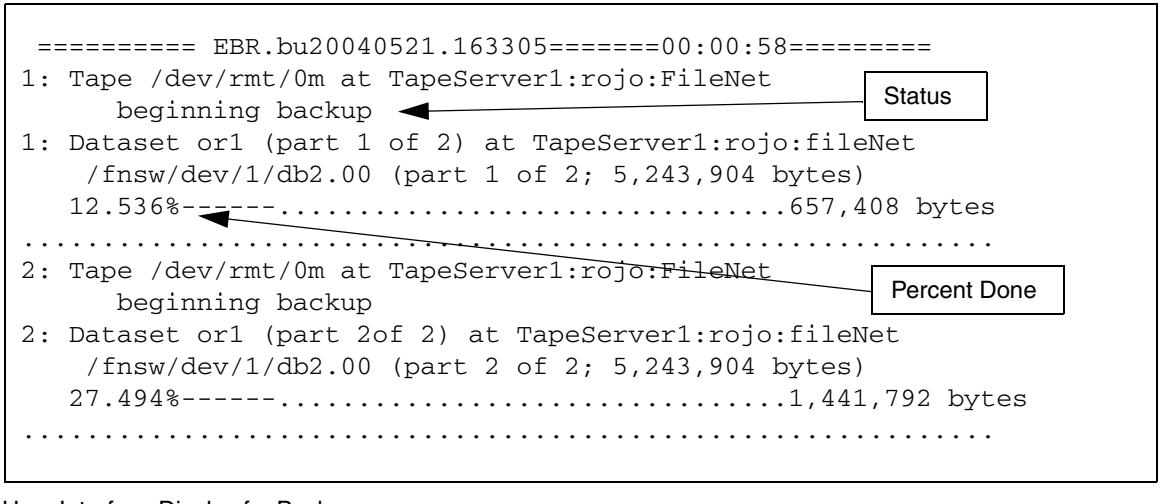
EBR normally starts a backup immediately. By using the start_date and start_time global parameters in your script, you can schedule the

backup for a future time. However, even if the date is in the future, the EBR process starts and opens a user interface display window. Then EBR waits until the specified date and time to start the backup. To cancel the waiting process, enter Control+c and reply **yes** to the prompt to terminate the process.

Note The prompt to confirm termination of a waiting process may not display immediately. The delay can be up to 60 seconds.

You must have a shell window open for the EBR user interface display to present status and operator request messages.

Your script defines one thread for each tape cartridge involved in the backup. Each thread is represented by five lines in the user interface display.



User Interface Display for Backup

The first two lines display the current status of the tape and the last three lines display the current status of the part of the disk dataset being backed up.

The first line of tape information in the user interface display starts with the thread number followed by a colon. The first line of the disk dataset status also starts with the thread number followed by a colon.

Any operator action required for the tape is displayed in the two lines of tape information. For example, the tape needed, the system on which it is needed, and the drive into which the tape must be inserted is displayed until the correct tape is inserted in the drive. Once the tape is inserted in the drive, EBR recognizes it and continues automatically. No additional operator response is required during the backup operation.

EBR special messages (for example, tape mount messages, successful completion, and unsuccessful completion) in the user interface display are in capital letters.

In addition to the user interface display, EBR produces a detailed progress log for every backup that is run. For detailed information, see **“Progress Log” on page 67**. From information in the progress log, you can compute throughput of the disk, the network, and the tape drive. You can also see how much disk data was read and how much compressed tape data was sent over the network and to the tape drive (less data is sent over the network than is sent to the tape drive). For examples and methods for computing throughput, see **“Appendix K – Calculating Throughput” on page 425**.

Additional status information is written to the system error log and the EBR summary log. See **“Status Reporting” on page 63** for more information about each log.

Backup Procedure

This section explains backup procedures for a combined server system, a dual server system, and a multiserver system.

Combined Server Backup Procedure

Comb

Follow the steps below to perform a backup of your system.

1 Open a window.

A shell window must be open for the EBR user interface display to present status and tape request messages.

2 Insert tapes that you prelabeled with EBR_label into the tape drives.

- If your script specifies a single thread, insert the tape cartridge into the tape drive.
- If your script uses multiple threads and the threads do **not** share a tape drive, insert a tape cartridge in each tape drive.
- If your script uses multiple threads and the threads share a tape drive, insert the first tape cartridge into the shared tape drive.

3 For online backups, skip to **Step 5 on page 242**.

For offline backups, click on the Task Manager Stop button to shut down the FileNet software.

4 For offline backups, click on the Task Manager Backup button to place the FileNet software in backup mode.

5 From the command line, enter the EBR command.

6 Monitor the progress log and syslog for completion messages.

If backup completes normally, skip to **Step 7 on page 243**.

If backup fails, determine the cause of the error and reset your system.

Note EBR may issue commands instructing you to run the EBR_ulmk or EBR_orreset utility. See **"Canceling the Backup" on page 247** for details. Run these utilities **before** you reset your system.

a Run the EBR_ulmk or EBR_orreset utility as instructed by EBR.

b Click the Task Manager Stop button followed by the Backup button to reset your system.

c Re-run your backup script.

7 At backup completion, eject the backup tape.

8 If your backup was offline, click the Task Manager Restart button to restart the FileNet software.

9 Document pertinent information about the backup tape and store it in a safe place.

Dual Server Backup Procedure

This procedure assumes a dual server configuration consisting of a root/index server and one storage library server.

Tip In a dual server environment, you can create one script that defines the datasets, backup options, and thread options for both servers. Then you can run that script on either server.

Follow the steps below to perform a backup of your dual server system.

1 Open a window.

A shell window must be open for the EBR user interface display to present status and tape request messages.

2 Insert tapes that you prelabeled with EBR_label into the tape drives.

- If your script specifies a single thread, insert the tape cartridge into the tape drive.
- If your script uses multiple threads and the threads do not share a tape drive, insert a tape cartridge in each tape drive.
- If your script uses multiple threads and the threads share a tape drive, insert the first tape cartridge into the shared tape drive.

3 For online backups, skip to **Step 4 on page 244**. For offline backups, continue with this step.

a At the storage library server, click the Task Manager Stop button to shut down the FileNet software.

b At the root/index server, click the Task Manager Stop button followed by the Backup button.

c At the storage library server, click the Backup button.

4 From the command line at either the root/index or storage library server, enter the EBR command.

5 Monitor the progress log and syslog for completion messages.

- If backup completes normally, skip to **Step 6 on page 245**.
- If backup fails, determine the cause of the error and reset your system.

Note EBR may issue commands instructing you to run the EBR_ulmk or EBR_orreset utility. See **"Canceling the Backup" on page 247** for details. Run these utilities **before** you reset your system.

a Run the EBR_ulmk or EBR_orreset utility as instructed by EBR.

b Click the Task Manager Stop button followed by the Backup button to reset your system.

c Re-run your backup script.

6 At backup completion, eject the backup tape.

Tapes are automatically ejected by some tape drives.

7 If your backup was offline, restart FileNet software.

Click the Task Manager Restart button on the root/index server first and then the storage library server.

8 Verify the backup by running EBR_tdir.

Refer to **"EBR_tdir" on page 345**.

9 Document the backup tape and store it in a safe place.

Multiserver Backup Procedure

MultSv

A multiserver configuration typically consists of a root/index server and one or more additional servers (for example, storage library servers and application servers). The configuration can vary widely. Use the EBR_genscript tool to build an appropriate script based on your unique multiserver configuration.

The multiserver online backup procedure is the same as for dual server online backup (see **"Dual Server Backup Procedure" on page 243**). However, the offline backup procedure is slightly different from that of the dual server offline backup procedure. Stopping and restarting each server in the multiserver configuration must be done in the proper sequence, as shown in the table below:

Action	Procedure
Stop FileNet software	1 Click the Task Manager Stop button on each non-root server in the following order: application servers followed by storage library servers. 2 On the root/index server, click the Task Manager Stop button followed by the Backup button. 3 On each non-root server in any order, click the Task Manager Backup button.
Restart FileNet software	1 On the root server, click the Task Manager Restart button. 2 Click the Task Manager Restart button on each non-root server in the following order: storage library servers followed by application servers.

In general, you stop application servers first and restart them last. However, depending on the Image Services services running on your application server, you may need to change the sequence slightly. For example, if you run index services on an application server, restart that application server immediately after restarting the root/index server.

Canceling the Backup

You can cancel an EBR operation at any time by typing a Control+c key sequence (press and hold the control key on your keyboard followed by the **c** key). However, do not cancel a backup unless you have a very good reason for doing so. When you type Control+c, a confirmation prompt displays.

Especially for offline backups, a cancel operation or a failure during backup can leave one or more databases in an error state that requires them to be unlocked or reset. The error state can also occur for online backups of Oracle databases. If the error state occurs for MKF datasets, use EBR_ulmk to unlock the database. If the error state occurs after an online Oracle database backup failure or cancellation, use EBR_orreset to reset the Oracle database state to "normal." (See **"Appendix A – Programs and Utilities" on page 265** for more information about these utilities.)

7
Running The Restore Script

EBR performs restore operations according to the type of script passed to the program. In this chapter, we cover the following topics:

- The command to start the EBR restore process
- Preliminary steps you must perform before you start a restore
- Procedures to perform a basic restore procedure

If you mirror all your disks, you may never have to do a restore because of a disk crash. Disk mirroring is the first line of defense against disk crashes. See **“Disk Mirroring and RAID” on page 27** for more information about disk mirroring.

Restore Program Command Syntax

The command line parameters of the EBR program are as follows:

EBR [–syntax] @<script> [<param> ...] [@<script> [<param> ...]] ...

The **–syntax** parameter permits you to check the syntax of your script before actually running the script. See **“Testing Your Script Syntax” on page 235** for more information.

Precede the restore script name with an @ sign with no intervening spaces. The script file is an ASCII file that you create as described in **Chapter 5, “Developing Your Scripts,” on page 147**.

The script may contain parameters. The value of all parameters must be specified on the command line following the script name. The syntax is:

<parameter_name>=<value>

with no embedded spaces. In the script, a parameter name starts with $. On the command line, the $ is omitted.

For UNIX environments, enter the command to start the restore on the command line of an open X window. The following example includes command line parameters enclosed in double quotes:

EBR @test_res location="costa10:FileNet"

For Windows Server environments, enter the command to start the restore on the command line of a DOS shell window. For parameters that need to be within double quotes, you must insert a backslash character (\) before each double quote as shown in the following example:

EBR @test.res location=\"iqant6:FileNet\"

You can run a series of scripts sequentially by specifying another script name followed by its parameters. If errors are encountered while running any script, EBR terminates and does not run the remaining scripts.

Preparing for Restore

Prepare your restore script. Decide whether you have both full and interval backup tapes to restore and set the interval_restore_follows

option to the proper value (true or false) in your script. Collect all tapes that you will need for the restore operation.

Before running a restore, FileNet software on the systems being restored must be shut down and in restore mode and databases must be offline. Use the Task Manager Restore button to place the FileNet software in the proper state. If you are performing a restore in a dual server or multiserver environment, the servers must be shut down and restarted in a particular order as described in **"Dual Server Restore Procedure" on page 255** or **"Multiserver Restore Procedure" on page 257**.

Before performing a restore, determine which datasets are affected. To save time, restore only the affected datasets.

When restoring cache and the transient database, be aware that cache differs from the databases. Cache cannot be rolled forward to the last complete transaction. See **"Cache" on page 101** for more information.

You usually need to perform two restores—a full restore followed by an interval restore. The exception is the MKF transient database and cache, for which EBR supports only restores of full backups. If you follow good backup discipline and perform full backups weekly and interval backups daily, you have a full backup and an interval backup tape to restore. If a problem occurs within a day of the full backup and that day's interval backup tape has not yet been created, you only have the full backup tape to restore.

If you have performed interval backups since the full backup, restore the full backup tape followed by the **most recent** interval backup tape made.

Note Remember that it is unnecessary to restore all interval backup tapes after the full restore. Interval backups are cumulative so after you restore the full backup tape, restore only the **most recent** interval backup tape.

When restoring databases, your restore script indicates whether this is a full restore and whether or not an interval restore follows. If you make a mistake about whether an interval restore follows, EBR may fail. For example, if the script for the full restore says that no interval restore follows, EBR tries to roll the database forward to the last transaction. If an interval restore actually is to follow, the rollforward will probably fail due to a gap in the redo log sequence and you will have lost your chance to roll the database forward to the last transaction.

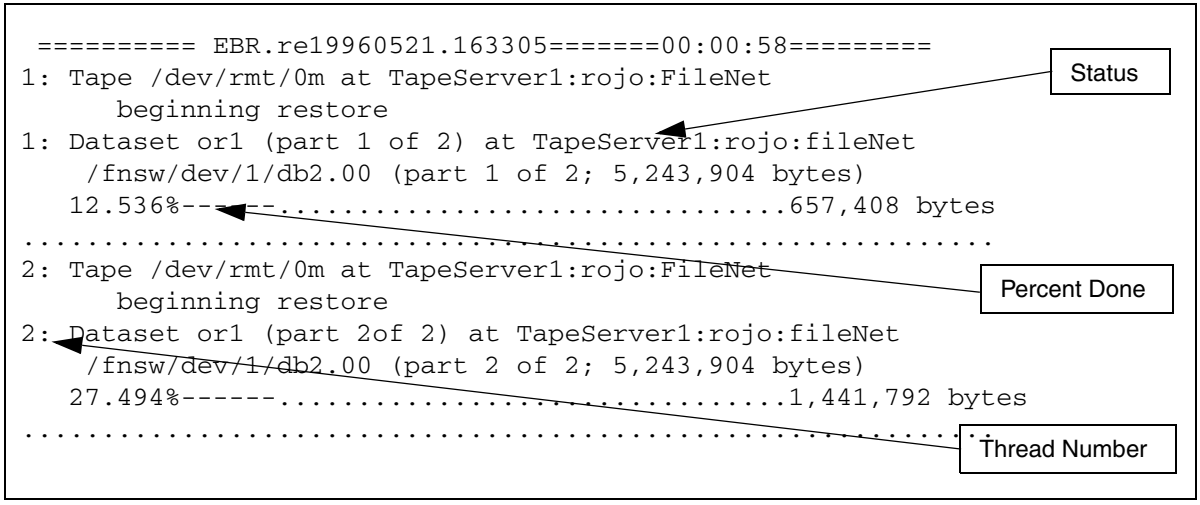
If an interval restore is done after the full restore, EBR attempts to roll the database forward to the last transaction only **after** the restore of the interval backup completes.

CAUTION The interval restore must occur **immediately** after the full restore completes. If the MKF database is opened normally before the interval restore is performed, EBR will be unable to roll the database forward to the last transaction. Restarting the FileNet software opens the MKF databases, so do not restart FileNet software until the interval restore completes.

A shell window must be open for the EBR user interface display to present status and operator request messages. (See also **"User Interface Display" on page 64**.)

Your script defines one thread for each tape cartridge involved in the restore. Each thread is represented by five lines in the user interface display. The first two lines display the current status of the tape and the

last three lines display the current status of the part of the magnetic disk dataset being restored.



User Interface Display for Restore

The first line of tape information in the user interface display starts with the thread number followed by a colon. The first line of the disk dataset status also starts with the thread number followed by a colon.

Any operator action required for the tape is displayed in the two lines of tape information. For example, the tape needed, the system on which it is needed, and the drive into which the tape must be inserted is displayed until the correct tape is inserted in the drive. Once the tape is inserted in the drive, EBR recognizes it and continues automatically. No additional operator response is required during the restore operation.

EBR special messages (for example, tape mount messages, successful completion, and unsuccessful completion) in the user interface display are in capital letters.

In addition to the user interface display, EBR produces a detailed progress log for every restore. For detailed information, see **"Progress Log" on page 67**.

An Oracle interval restore operation restores only the data changed since the last full backup. In the progress log, EBR reports only the number of changed bytes that were restored, not the total number of bytes in the database.

From information in the progress log, you can compute throughput of the disk, the network, and the tape drive. You can also see how much disk data was read and how much compressed tape data was sent over the network and to the tape drive (less data is sent over the network than is sent to the tape drive). For examples and methods for computing throughput, see **"Appendix K – Calculating Throughput" on page 425**.

Additional status information is written to the system error log and the EBR summary log. For more information about each of these logs, see **"Status Reporting" on page 63**.

Restore Procedure

This section explains restore procedures for a combined server system, a dual server system, and a multiserver system.

Combined Server Restore Procedure

Comb

Follow the steps below to restore your system.

1 Open a window.

A shell window must be open for the EBR user interface display to present status and operator request messages.

2 Insert tapes in the tape drives.

- If your script specifies a single thread, insert the tape cartridge into the tape drive.
- If your script uses multiple threads and the threads do not share a drive, insert a tape cartridge in each drive.
- If your script uses multiple threads and the threads share a drive, insert the first tape cartridge into the shared drive.

3 Click on the Task Manager Restore button to place the FileNet software in restore mode.

4 Enter the EBR command at the command line.

5 Monitor the progress log and syslog for completion messages.

- If restore completes normally, skip to **Step 6**.
- If restore fails, determine the cause of the error and reset your system.

Note EBR may issue commands instructing you to run the EBR_ulmk or the EBR_orreset utility. See **"Canceling the Restore" on page 264** for details. Run these utilities **before** you reset your system.

a Click the Task Manager Restore button to reset your system.

b Re-run your restore script.

6 When restore completes, eject the backup tape and check the progress and summary logs for successful completion messages.

If an MKF permanent database restore did not successfully roll forward, see **"Scalar Numbers Table Update After Restore Failure" on page 259**.

7 Click the Task Manager Restart button to restart the FileNet software.

8 Back up your newly restored system.

Dual Server Restore Procedure

This procedure assumes a dual server configuration consisting of a root/index server and one storage library server. Follow the steps below to perform a restore of your dual server system.

1 Open a window.

A shell window must be open for the EBR user interface display to present status and operator request messages.

2 Insert tapes in the tape drives.

- If your script specifies a single thread, insert the tape cartridge into the tape drive.

- If your script uses multiple threads and the threads do not share a drive, insert a tape cartridge in each drive.
- If your script uses multiple threads and the threads share a drive, insert the first tape cartridge into the shared drive.

3 Place the FileNet software on the servers in restore mode in the following order:

a At the storage library server, click the Task Manager Stop button to shut down the FileNet software.

b At the root/index server, click the Task Manager Stop button followed by the Restore button.

c At the storage library server, click the Restore button.

4 At the command line of either the root/index or storage library server, enter the EBR command.

5 Monitor the progress log and syslog for completion messages.

- If restore completes normally, skip to **Step 6**.
- If restore fails, determine the cause of the error and perform the following steps to reset your system.

Note EBR may issue commands instructing you to run the EBR_ulmk or the EBR_orreset utility. See **"Canceling the Restore" on page 264** for details. Run these utilities **before** you reset your system.

a Click the Task Manager Restore button to reset your system.

b Re-run your restore script.

6 When restore completes, eject the backup tape and check the progress and summary logs for successful completion messages.

If an MKF permanent database restore did not successfully roll forward, see **"Scalar Numbers Table Update After Restore Failure" on page 259**.

7 Restart the FileNet software on each server in the following order:

a At the root/index server, click the Task Manager Restart button.

b At the storage library server, click the Task Manager Restart button.

8 Back up your newly restored system.

Multiserver Restore Procedure

MultSv

A multiserver configuration typically consists of a root/index server and one or more additional servers (for example, storage library servers and application servers). The configuration can vary widely. Use the EBR_genscript tool to build an appropriate script based on your unique multiserver configuration.

Stopping and Restarting Each Server

The multiserver restore procedure is the same as for dual server restore (see **"Dual Server Restore Procedure" on page 255**). The

stopping and restarting each server in a multiserver configuration must be done in the proper sequence, as shown in the table below:

Action	Procedure
Stop FileNet software	1 Click the Task Manager Stop button on each non-root server in the following order: application servers followed by storage library servers. 2 On the root/index server, click the Task Manager Stop button followed by the Backup button. 3 On each non-root server in any order, click the Task Manager Backup button.
Restart FileNet software	1 On the root server, click the Task Manager Restart button. 2 Click the Task Manager Restart button on each non-root server in the following order: storage library servers followed by application servers.

In general, you stop application servers first and restart them last. However, depending on the Image Services running on your application server, you may need to change the sequence slightly. For example, if you run index services on an application server, restart that application server immediately after restarting the root/index server.

Procedure for a Visual WorkFlo Database

Perform this additional step only if you have a multi-server system where your Oracle database contains a **Visual WorkFlo** database.

After you restart the Image Services when the restore completes, run the **VW Verify** utility to check and reconcile any discrepancies within the Visual WorkFlo database. The default VW Verify command checks the Visual WorkFlo database and reports discrepancies. On the client workstation or on the server, enter: **vwverify**

To report and, if confirmation is received, repair discrepancies, enter:

vwverify –f

After an Oracle database restore in a multi-server system, regardless of whether one, a few, or all servers are restored, the servers may not be synchronized with one another because one server may have parts of a transaction which another server does not have. Therefore, after the restore, run the **VW Verify** utility. VW Verify will find and fix more discrepancies if not all servers are restored than if all servers are restored, but VW Verify will work in both cases. For details on running the VW Verify utility, refer to the *Visual WorkFlo Installation and Administration Handbook, Release 3.0.*

Note We recommend restoring all servers if transfer has been run since the backup and server 0 must be restored. The reason is that server 0 contains work and work performer class definitions. If only server 0 is restored and the class definitions are erased, then work objects on other servers which reference these class definitions will not be usable.

If you restore only server 0, then you must run VW Verify to delete all the work objects which no longer have class definitions on server 0. This deletion process may take significantly longer than restoring all servers.

Scalar Numbers Table Update After Restore Failure

If a restore of the MKF permanent database successfully rolled the database forward to the last complete transaction, you are finished with the restore.

If the permanent database was not successfully rolled forward, you must use the SNT_update program to update the scalar numbers table before you continue. For example, the following sequence of messages display when the MKF databases cannot be rolled forward due to recovery log corruption:

```
MKF db: /fnsw/dev/1/sec_db0
The above database was just restored, and the recovery log on disk
did not splice in, so all updates since the backup was created, if any,
will be lost:  Database can not be rolled forward.
02/05/02 16:36:10.742 161,0,1335 <fnsw> BRTs (1449) ...
NOT AN ERROR -- Informational message only:
MKF db: /fnsw/dev/1/permanent_db0
The above database was just restored, and the recovery log on disk
did not splice in, so all updates since the backup was created, if any,
will be lost:  Database can not be rolled forward.
02/05/02 16:46:47.610 161,0,1335 <fnsw> BRTs (1487) ...
NOT AN ERROR -- Informational message only:
MKF db: /fnsw/dev/1/transient_db0
The above database was just restored, and the recovery log on disk
did not splice in, so all updates since the backup was created, if any,
will be lost:  Database can not be rolled forward.
02/05/02 16:46:53.204  <fnsw> BRTs (1489) ...
CSMl(1489): Cache 1 successfully initialized
02/05/02 16:46:53.315 63,0,10 <fnsw> BRTs (1489) ... [CRITICAL]
The Scalar Numbers Table is behind the snt.chkpt file.  This should
 only happen after a Permanent DB  restore has been done. Continuing
 with this condition may cause multiple documents to be committed
 with the same doc ID. To resolve this problem you must update the
 Scalar Numbers Table with the SNT_update program.  Doc Services
 will not function until this problem is resolved.
```

You **must** run the SNT_update program to ensure that duplicate document numbers are not assigned when document entry resumes.

CAUTION The SNT_update program must be run before you attempt to restore cache and the transient database.

A checkpoint file, snt.chkpt, acts as a backup to the scalar numbers table. The checkpoint file is located in the following directory:

UNIX /fnsw/local/sd/snt.chkpt for UNIX systems

WIN <drive>:\fnsw_loc\sd\snt.chkpt for Windows Server systems

FileNet software updates this checkpoint file with current document numbers as documents are written to optical storage media. Running SNT_update after a restore updates the restored snt.chkpt with values from a more current saved copy of the snt.chkpt file. (See **"Appendix H – Running SNT_update" on page 387**.)

Note If the current snt.chkpt file is lost or corrupted, contact your service representative for assistance.

After SNT_update completes, document entry of any lost documents can proceed. After these documents are rescanned, reentered, reassigned document numbers, and recommitted, datasets on both magnetic media and storage media will be synchronized.

Updating the Configuration Database File after Restore

This section describes how to update the configuration database file after you complete a restore operation.

Note If you have not changed your FileNet system configuration since the last backup, you do not need the information in this section.

EBR does not back up or restore file systems or subdirectory trees. This means that the FileNet configuration database (CDB) files are not included on EBR media. For files not supported by EBR you must use another backup/restore method such as an operating system copy utility or a third-party backup program.

The path name of a configuration database file is:

UNIX

/fnsw/local/sd/conf_db/IMS_*nnn*.cdbfor UNIX platforms

WIN

<drive>:\fnsw_loc\sd\conf_db\IMS_*nnn*.cdbfor Windows Server platforms

where *nnn* is the file number. The highest numbered CDB file reflects the current configuration of your FileNet system. Your system configuration state and the current CDB file should be consistent.

You will need to copy or rename the restored CDB file into the next highest numbered CDB file. You should then update the CDB file to reflect the system changes you entered since the last backup (instructions will follow).

Perform this step if you have changed your FileNet configuration database (CDB) file since the last backup, especially if the change was significant. This will ensure that your system state and the current CDB file are consistent.

Significant changes include adding new datasets or partitions. Other changes include adding system devices such as optical devices, printers, tape drives and changing tuning parameters such as memory buffers and disk space allocations

NOTE Your configuration database file (CDB) changes when you run **fn_edit** (or Configuration Editor on NT) and save your changes.

The reason this step is included is that the restored system state may not match the current CDB file used by the Configuration Editor. The CDB file number is advanced to the next highest number each time you update your FileNet system configuration through the fn_edit utility. (Older files are not automatically deleted.) When you restart your system after a full restore, the Configuration Editor uses the highest numbered CDB file found in the directory.

For example, at the time you performed the last full backup, the CDB file number was 10 (IMS_10.cdb). Later you ran fn_edit to add extents and datasets. This action created and the current CDB file: IMS_11.cdb. After a full restore, the CDB file that is currently recognized by the system is IMS_10.cdb **not** IMS_11.cdb. IMS_11.cdb was overwritten in the restore and therefore, IMS_10.cdb has the highest number. You should copy IMS_10.cdb into the next highest numbered file, IMS_12.cdb.

Steps To Update The CDB File

1. Copy the CDB file by using the UNIX **cp** command or the Windows Server **copy** command (or File Windows Server Explorer method).

2. Use the fn_edit utility (or Configuration Editor on Windows Server) and re-enter changes made to the configuration since the last backup.

3. Synchronize the configuration files and the datasets by entering the following on the command line:

   fn_build -a

Canceling the Restore

You can cancel an EBR restore operation at any time using the Control+c key sequence (press and hold the control key on your keyboard followed by the c key). However, do not cancel a restore unless you have a very good reason for doing so. After canceling a restore, the dataset being restored is unusable until a subsequent restore successfully completes.

When you use Control+c, a confirmation prompt displays. If you are sure you want to proceed with the cancellation of the restore operation, reply **yes** to the prompt.

A cancel operation or a failure during restore can leave one or more databases in an error state that requires them to be unlocked or reset. If the error state occurs for MKF datasets, use EBR_ulmk to unlock the database. If the error state occurs for Oracle databases, use EBR_orreset to reset the Oracle database state. See **“Appendix A – Programs and Utilities” on page 265** for more information about these utilities or contact your service representative for assistance.

Restoring an Oracle Dataset to a Different System

If you need to restore an Oracle dataset to a different system, see the procedures in **“Appendix J – Restoring Oracle” on page 411**.

Appendix A – Programs and Utilities

In this appendix, we provide a description of the programs and utilities that make up Enterprise Backup/Restore. For information about how EBR works internally to accomplish its automated backup and restore operations, see **“Appendix C – EBR Program Operation” on page 363**.

The following Enterprise Backup/Restore programs and utilities are described in this appendix:

EBR

The EBR program runs in a shell window and performs backups and restores. The command line syntax to start EBR is:

EBR [–syntax] @<script> [<param> ...] [@<script> [<param> ...]] ...

The **–syntax** parameter, if specified, appears before @<script> and checks the script syntax without running the script. See **"Testing Your Script Syntax" on page 235** for more information.

Specify the name of your backup or restore script file in <script>. Spaces are not allowed between @ and the script file name. The script file is an ASCII file (described in **Chapter 5, "Developing Your Scripts," on page 147**). You can specify multiple scripts in the same command. Scripts are run sequentially in the order encountered in the command.

If you run multiple scripts sequentially from the same command, information in the user interface display for the first script is overwritten when the second script runs, and so on. For information about each script, see the appropriate progress log. If you don't know the progress log file name, you can use an operating system command to list all the progress logs for a particular day. For example, the following UNIX command lists all the backup progress logs created on May 1 of the current year:

```
ls  /fnsw/local/logs/EBR/*bu*0501*
```

Use File Manager in Windows Server to list progress logs. Use Notepad or an editor of your choice to examine the contents of a progress log in the Windows Server environment.

Your EBR command may include parameters. You must specify the value of all parameters on the command line following the script name. The syntax is:

```
<parameter_name>=<value>
```

If the value is a quoted string on a UNIX platform, the syntax is:

‘<parameter_name>=“<value>” ’

If the value is a quoted string on a Windows Server platform, the syntax is:

<parameter_name>=\“<value>\”

No embedded spaces are allowed in any of the syntax formats. In your script, start the parameter name with a $ sign. On the command line, omit the $ sign.

Note If the value is a quoted string, such as the host location name, use single apostrophes and double quotes appropriately around parameter names and their values as shown above. The shell automatically strips off double quotes. However, double quotes are required as part of the parameter value. Enclosing command line parameters within single apostrophes prevents the shell from stripping off the double quotes. The apostrophes will instead be stripped off by the shell, leaving the required double quotes in place.

Command Line Help

If you run EBR with no parameters or illegal parameters, extensive help text displays to stdout (the standard output device, usually your shell window).

EBR Command Example

The following command entered at the shell prompt runs a script called quickbu.tape. If you cannot fit the entire command on a single line, use backslash (\) command line continuation characters as shown in the example:

```
EBR @quickbu.tape \
    'disk_serv="TapeServer1:bandit:FileNet" ' \
    'tape_serv="TapeServer1:bandit:FileNet" '
```

EBR_clean

The EBR_clean utility is for use **only by IBM support personnel** to reclaim shared memory and interlocks for a failed backup or restore.

CAUTION If you start EBR_clean on a host on which a backup or restore is running, the backup or restore fails and the system may hang.

If an interlock is claimed by a dead EBR process, EBR_clean will hang as well. To release an interlock claimed by a dead EBR process, recycle the FileNet software or reboot the server.

EBR_genscript

The EBR_genscript tool generates dataset definition files and builds backup and restore scripts through a question-and-answer interface. In addition, EBR_genscript optionally determines the number of tapes required to back up the selected datasets.

Note Before using EBR_genscript to create production-level scripts, you must be thoroughly familiar with EBR terminology and the backup and restore strategy for your enterprise. In addition, you must be familiar with EBR script syntax. You must review the output of EBR_genscript to verify correct backup and restore logic.

The dataset definition file or generated script is syntactically correct. However, a generated script is not guaranteed to be logically correct. For example, you can specify non-FileNet datasets and the correct

syntax is produced but EBR_genscript does not generate order constraints for non-FileNet datasets.

At a minimum, you should read the following chapters in this manual:

Chapter 2, “Understanding EBR Concepts,” on page 42
Chapter 4, “Developing Your Backup Strategy,” on page 89
Chapter 5, “Developing Your Scripts,” on page 147

FileNet Datasets

EBR_genscript automatically locates all FileNet datasets for any selected domain. You only need to specify a server name in a domain and EBR_genscript locates all FileNet datasets in the selected domain and establishes the correct order constraints for these datasets. FileNet dataset types are MKF databases (security, transient, and permanent), index database (managed by Oracle), and cache.

Generic Datasets

Using EBR_genscript, you can also include non-FileNet-specific (“generic”) datasets in your EBR backup/restore strategy. Non-FileNet dataset examples are raw partitions and Oracle databases that are not configured by FileNet configuration tools (and thus not defined in the configuration database). These databases are also referred to as “site-controlled” in a database coexistence environment. For generic datasets, you must specify the host first and then the dataset type. You must also provide information about the dataset such as base data file name, dataset size, and, for partitions, the starting block and blocksize.

The tool prompts you for information and builds the script or dataset definition file based on your responses. For certain dataset types (partition, MKF transient database, and cache), EBR_genscript automatically sets the type and mode of backup to the proper options (that is, full backup in offline mode).

Starting EBR_genscript

To start the tool, enter **EBR_genscript** at the command line.

Tip Output files (.ddf files and scripts) are written to the current directory, that is, the directory that is current when you start EBR_genscript. If you want to change the current directory, do so before starting the tool. If you want your output files to be written to a directory other than the current directory, you must supply the full path name of each file when responding to prompts.

The main menu displays:

```
Mars(user1)/fnsw/local> EBR_genscript

===== EBR_genscript Main Menu =====

  [1] - Generate dataset definitions
  [2] - Generate device specification
  [3] - Generate backup script
  [R] - Generate restore script

  [9] - Help
  [0] - Exit

Enter a command ==>
```

Getting Help

Help is available for each option in the main menu. Enter **9** to display the following screens:

```
===== EBR_genscript Main Menu =====

  [1] - Generate dataset definitions
  [2] - Generate device specification
  [3] - Generate backup script
  [4] - Generate restore script

  [9] - Help
  [0] - Exit

Enter a command ==> 9

[1] Generate datasets definition:
    This option generates dataset definitions for the Enterprise Backup/
Restore utility.

[2] Generate device specification:
    This option generates device specification for the Enterprise Backup/
Restore utility.

[3] Generate backup script:
    This option generates backup scripts using pre-defined datasets
generated by option [1].

[4] Generate restore script:
    This option generates restore scripts using pre-defined datasets
generated by option [1].

(continued on next page)
```

```
(continued from previous page)

[9] Help:
    This menu

[0] Exit:
    Exit current menu

When generating dataset definitions, backup scripts, and restore scripts,
script variables can be used as substitute for specifying parameters.
These script variables are substituted during run time.  A script variable
always starts with a dollar sign ($).
-- End -- (Enter CR to continue)
```

Recommended EBR_genscript Steps

The recommended steps for running EBR_genscript are the following:

1. Complete the EBR_genscript worksheet. See **"EBR_genscript Worksheets" on page 273**.
2. Generate a dataset definitions file by entering the **1** option from the Main Menu. Refer to **"Creating a Dataset Definition File" on page 279** for detailed procedures and examples.
3. Generate a device specification file by entering the **2** option from the Main Menu.
4. Generate a backup script using the dataset definitions file you created in Step 2. Refer to **"Creating a Backup Script" on page 305** for detailed procedures and examples.

Depending on your backup strategy, you may need to run EBR_genscript multiple times to develop separate scripts for interval (nightly) backups and full (weekly) backups.

5 Generate a restore script based on the dataset definition file and the information you provided for the backup script in Step 2. Refer to **"Creating a Restore Script" on page 319** for detailed procedures and examples.

EBR_genscript Worksheets

Before you run EBR_genscript, we recommend that you develop worksheets similar to those in this section. Complete the worksheets. Use the answers from your worksheets to respond to EBR_genscript prompts.

The following worksheets can be used as examples for developing your own.

Dataset Definition Worksheet

- Dataset definition file name: _____
- Dataset type to back up:
  - _____ FileNet
    - _____ Local domain
    - _____ Remote domain
    - For FileNet datasets in a **remote** domain, host name of a server in the domain: _____

- _____ Generic (non-FileNet)

- For FileNet datasets, location of Oracle signature file directory:

  For a local domain _____

  For a remote domain _____

- For generic (non-FileNet) datasets:

  Host name: _____

  For Oracle dataset:

  - Oracle dataset name: _____
  - Oracle dataset size (estimated): _____
  - Oracle signature file directory: _____
  - Oracle parameter file name: _____

- For partition dataset:

  Partition dataset name: _____

  Full path name for partition: _____

  Partition block size (in bytes): _____

  Partition size in blocks (or 0 for entire partition): _____

  Partition size (in bytes): _____

Backup Script Worksheet

- Backup script file name: _____
- Volume group name: _____
  _
- Backup media expiration (days): _____
- Dataset definition file (.ddf) directory: _____
- Dataset definition file (.ddf) name: _____
- Device specification file directory: _____
- Device specification file name:_____
- List of dataset names to back up_____
- # of threads to use for backup: _____

Depending on the choices you make, the paths through EBR_genscript vary. Based on your backup strategy and system configuration, create a table containing information for which EBR_genscript will prompt you. For example, for each dataset on each host, your table could include the following:

- Archive log retention (in days)
- Backup mode (online or offline)
- Number of stripes (if applicable) to be used for each dataset and which stripe is assigned to which thread
- Host providing tape service for each thread
- Volume serial number for each thread

- Backup media and device information
  - Disk
  - Tape
    - Tape device type (8 mm, 4 mm, QIC)
    - Tape rewind/no-rewind device names
    - Tape capacity (in bytes)
  - Tape Library
    - Tape library device type (EXB-210, EXB-218, EXB-220)
    - Tape library device (robotic arm) filename
    - Tape drive number
    - Tape info (all information under Tape bullet above)

  If you plan to use more than one thread, develop a thread assignment table. You can include the backup mode and type for each dataset in this table, as shown below:

Dataset-to-Thread Assignment for Backup

				Backup Mode		Backup type	
Thread Number	**Host Name**	**Dataset Name**	**Stripe n of m**	**Online**	**Offline**	**Full**	**Interval**

Dataset-to-Thread Assignment for Backup

				Backup Mode		Backup type	

Restore Script Worksheet

The restore worksheet is similar to the backup worksheet. Modify the following example worksheet to meet your needs.

- Restore script file name: _____
- Volume group name: _____
- Dataset definition file (.ddf) directory: _____
- Device specification file directory:_____
- # of threads to use for restore: _____
- If you plan to use more than one thread, develop a dataset-to-thread assignment table similar to that in the backup worksheet.
- Host providing tape service: _____
- Volume serial # for thread 1: _____
- Backup media type:

  _____ Tape

  _____ Disk

  _____ Tape library

- Tape type:

  _____ 8 mm

  _____ 4 mm

  _____ QIC

- Tape rewind device name: _____
- Tape no-rewind device name: _____
- Tape library type:

  _____ EXB-210

  _____ EXB-218

  _____ EXB-220

- Tape library device (robotic arm) filename_____
- Tape library drive number _____
- Tape library tape rewind/no-rewind device names
  _____
- Tape library tape device type (8 mm, 4 mm, QIC)
  _____
- Tape library tape capacity (in bytes) _____

Creating a Dataset Definition File

To generate a dataset definition file, enter **1** from the main menu:

```
== EBR_genscript Main Menu

  [1] - Generate dataset definitions
  [2] - Generate Device Specification
  [3] - Generate backup script
  [4] - Generate restore script
  [9] - Help
  [0] - Exit

Enter a command ==> 1
Enter EBR dataset definition file name
(suffix '.ddf' will be appended to the file name
entered)

==> /fnsw/local/EBR/tt
```

EBR_genscript prompts you for the dataset definition file name and appends the .ddf extension to the file name. If the file name already exists, a message similar to the following displays:

‘/fnsw/local/EBR/tt.ddf’ exists, overwrite [y/n]?

Reply y(es) to overwrite the existing file or n(o) to specify a new file.

A series of menus follows for defining FileNet datasets and generic datasets in local and remote domains. FileNet dataset definitions, such as MKF datasets and the Oracle signature file, are required; generic dataset definitions, such as partitions, are optional.

You can generate a dataset definitions file for the simplest case: FileNet datasets in a single local domain. You can generate separate

dataset definitions files for additional local domains and remote domains. On the other hand, you can generate a single .ddf file that includes multiple domains (local and remote) and includes both FileNet and non-FileNet (generic) datasets.

Note EBR allows inclusion of multiple dataset definition files in a backup or restore script. However, EBR_genscript allows inclusion of only one dataset definition file when you generate a backup or restore script. To include multiple dataset definition files in the backup or restore script generated by EBR_genscript, edit the .bac or .res file to add an #include statement for each dataset definition file you want to include.

EBR_genscript dataset definition file examples in this section illustrate both simple and complex options. The dataset definition file generated by responses are shown after each example. Your dataset definition file will be different.

Defining FileNet Datasets for a Single Local Domain

The following examples show a typical procedure for defining FileNet datasets on the local domain (the most common situation). In this example, “Mars” is the local domain name:

```
=====  Define Dataset (Menu 1.0) =====

  [1] - FileNet datasets (required)
  [2] - Generic datasets (optional)

  [9] - Help
  [0] - Done

  Enter a command ==> 1

===== Define FileNet Dataset (Menu 1.1) =====

  [1] - Define datasets for local default domain
  [2] - Define datasets for remote domain
  [3] - Browse all selected domains
  [9] - Help
  [0] - Done

  Enter a command ==> 1

Enter the full path name of Oracle signature file
directory on host 'Mars'.
The signature directory should have enough disk space
to hold
at least 1/32 of the size of the database.
==> /fnsw/local/tmp/ora_sig
```

Note The recommended location for the Oracle signature file is /fnsw/local/tmp; however, this location is not required.

Your response to the prompt for the Oracle signature file location completes the FileNet dataset definition for the local domain. The “Define FileNet Dataset” menu redisplays with a highlighted confirmation message, as shown below:

```
===== Define FileNet Dataset (Menu 1.1) =====

  [1] - Define datasets for local default domain
  [2] - Define datasets for remote domain
  [3] - Browse all selected domains
  [9] - Help
  [0] - Done
  Successfully defined datasets for local domain

  Enter a command ==>
```

You can browse your selections, continue with dataset definition for a remote domain, or enter **0** to complete this task.

Tip If you enter **Done** and then Exit to exit, EBR_genscript completes the .ddf file creation. You cannot run EBR_genscript again to append additional datasets to an existing .ddf file. Each time you start EBR_genscript, the existing .ddf file is either overwritten or a new .ddf is created. To add to or change an existing .ddf file, you must manually edit the file with a text editor.

At the main menu, you can enter options to create a backup or restore script, or, if you do not wish to generate a script at this time, enter **0** to exit EBR_genscript.

Below is the output (tt.ddf) generated by the example on the previous pages.

```
DATASETS
------------------- Domain             Mars:FileNet -------------------
--  Mars             HP9000 : Server ID: 1, Server Type : Combined
-------------------------------------------------------------------------

-- Begin Datasets for Server 'Mars' of 'Mars:FileNet' -----------------

  Oracle_Mars : Oracle
--  DatasetSize = 314,572,800
    location = "Mars";
    signature_file_directory = "/fnsw/local/tmp/ora_sig";
  end_Oracle

PermDB_Mars : MKF
--  DatasetSize = 146,800,640
    location = "Mars";
    base_data_file = "/fnsw/dev/1/permanent_db0";
  end_MKF

  SecDB_Mars : MKF
--  DatasetSize = 16,777,216
    location = "Mars";
    base_data_file = "/fnsw/dev/1/sec_db0";
  end_MKF

  TranDB_Mars : MKF
--  DatasetSize = 62,914,560
    location = "Mars";
    base_data_file = "/fnsw/dev/1/transient_db0";
    transient_db;
  end_MKF

  PageCache_Mars : cache
--  DatasetSize = 83,886,080
    location = "Mars";
```

```
    transient_db = TranDB_Mars;
    permanent_db = PermDB_Mars;
    security_db = SecDB_Mars;
  end_cache
 -- End Datasets for Server 'Mars' of 'Mars:FileNet' -------------------

--------------------- Order Constraints ------------------------------
  order_constraints
    secDB_mars, PermDB_mars, TranDB_mars before PageCache_mars;
  end_order_constraints
END_DATASETS
```

Defining FileNet and Generic Datasets for a Single Local Domain

You can create a single .ddf file that includes both FileNet and non-FileNet (generic) datasets from the local domain. The following

example generates a dataset definition file that includes the required FileNet datasets and a partition.

```
=====  Define Dataset (Menu 1.0) =====

  [1] - FileNet datasets (required)
  [2] - Generic datasets (optional)

  [9] - Help
  [0] - Done

  Enter a command ==> 1

===== Define FileNet Dataset (Menu 1.1) =====

  [1] - Define datasets for local default domain
  [2] - Define datasets for remote domain
  [3] - Browse all selected domains
  [9] - Help
  [0] - Done

  Enter a command ==> 1

Enter the full path name of Oracle signature file
directory on host 'Mars'.
The signature directory should have enough disk space
to hold at least 1/32 of the size of the database.
==> /fnsw/local/tmp/ora_sig
```

```
===== Define FileNet Dataset (Menu 1.1) =====

  [1] - Define datasets for local default domain
  [2] - Define datasets for remote domain
  [3] - Browse all selected domains
  [9] - Help
  [0] - Done
  Successfully defined datasets for local domain

  Enter a command ==>
```

After the confirmation message for the required datasets, you can optionally define generic datasets, which include partitions and site-controlled Oracle databases.

Enter 0 (Done) to return to the Define Dataset screen.

At the prompt for a new command, enter **2** to start generic dataset definition and, at the following prompt, enter the name of the host on which the partition resides:

```
=====  Define Dataset (Menu 1.0) =====

  [1] - FileNet datasets (required)
  [2] - Generic datasets (optional)
  [9] - Help
  [0] - Done

  Enter a command ==> 2

  Enter host ==> mars

===Define Generic Dataset for Host 'mars' (Menu 1.2)===

  [1] - Define partition dataset
  [2] - Define Oracle database dataset

  [9] - Help
  [0] - Done

  Enter a command ==> 9
```

Help for each option on the generic definition menu is available. Enter **9** at the prompt for “Define Generic Dataset for Host” menu to display the following help text:

```
This menu helps users to define generic dataset types
such as Oracle database, partition, file, and
directory.
Note: directory and file have not been implemented.

When defining a generic dataset, you must provide the
dataset size.  The size information will be used to
estimate the required tape capacity.

[1] Define partition dataset:
    You must provide the partition size in bytes.

[2] Define Oracle database dataset:
    Before defining Oracle database dataset, you must
    provide the total size, in bytes, that Oracle
    database uses.  Total size equals the size of all
    Oracle data files, online redo logs, and rollback
    segments.

[0] Done:
    Exit current menu
-- End -- (Enter <CR> to continue)
```

Enter **1** to begin partition definition; enter **2** to begin Oracle database dataset definition.

In the following example, a partition is defined by entering **1**:

```
===Define Generic Dataset for Host 'mars' (Menu 1.2)===

  [1] - Define partition dataset

  [9] - Help
  [0] - Done

  Enter a command ==> 1

Enter partition dataset name ==> part

Enter the full path name for the partition filename
Example : /dev/hd2
==> /dev/vg00/fnsw
Enter block size in bytes ==>   1024
Enter partition size in blocks, or 0 for the entire
partition
==> 0

If you know the partition size, enter partition size
now.
The partition size will help to estimate tape capacity.
If you don't know the partition size, enter <CR>.
Enter partition size in bytes ==> <cr>
```

Note In the example above, the partition size is unknown so a carriage return is used to respond to the “Enter partition size in bytes” prompt.

When the partition definition is complete, enter **0** (Done) to exit this section. Continue to exit until you reach the EBR_genscript main menu.

```
===Define Generic Dataset for Host 'mars' (Menu 1.2)===

  [1] - Define partition dataset

  [9] - Help
  [0] - Done
  Successfully defined partition dataset on host 'mars'

  Enter a command ==> 0
```

At the main menu, you can enter options to generate a device specification file and to create a backup or restore script or enter **0** to exit EBR_genscript.

The .ddf file generated from this example is shown below:

```
DATASETS
------------------- Domain             Mars:FileNet -------------------
--  Mars             HP9000 : Server ID: 1, Server Type : Combined
-----------------------------------------------------------------------

-- Begin Datasets for Server 'Mars' of 'Mars:FileNet' -----------------

  Oracle_Mars : Oracle
--  DatasetSize = 314,572,800
    location = "Mars";
    signature_file_directory = "/fnsw/local/tmp/ora_sig";
  end_Oracle

  PermDB_Mars : MKF
--  DatasetSize = 146,800,640
```

```
    location = "Mars";
    base_data_file = "/fnsw/dev/1/permanent_db0";
  end_MKF

  SecDB_Mars : MKF
--  DatasetSize = 16,777,216
    location = "Mars";
    base_data_file = "/fnsw/dev/1/sec_db0";
  end_MKF

  TranDB_Mars : MKF
--  DatasetSize = 62,914,560
    location = "Mars";
    base_data_file = "/fnsw/dev/1/transient_db0";
    transient_db;
  end_MKF

  PageCache_Mars : cache
--  DatasetSize = 83,886,080
    location = "Mars";
    transient_db = TranDB_Mars;
    permanent_db = PermDB_Mars;
    security_db = SecDB_Mars;
  end_cache

-- End Datasets for Server 'Mars' of 'Mars:FileNet' --------------------

-- Begin Generic Datasets for host 'mars' -------------------

  part : partition
    location = "mars";
    filename = "/dev/vg00/fnsw";
  end_partition

-- End Generic Datasets for host 'mars' -------------------
```

Defining FileNet and Generic Datasets for Multiple Domains

A single dataset definition file can include FileNet and non-FileNet datasets from the local and one or more remote domains.

Note As noted earlier in this section, you can create separate .ddf files if you choose. EBR allows inclusion of multiple .ddf files. However, EBR_genscript is limited to the use of one .ddf file.

The following sequence of responses generates a .ddf file that includes FileNet datasets from the local domain (mars) and non-FileNet datasets from two remote domains (corona and venus):

```
=====  Define Dataset (Menu 1.0) =====

  [1] - FileNet datasets (required)
  [2] - Generic datasets (optional)
  [9] - Help
  [0] - Done

  Enter a command ==> 1

===== Define FileNet Dataset (Menu 1.1) =====

  [1] - Define datasets for local default domain
  [2] - Define datasets for remote domain
  [3] - Browse all selected domains
  [9] - Help
  [0] - Done

  Enter a command ==> 1

Enter the full path name of Oracle signature file
directory on host 'Mars'.
The signature directory should have enough disk space to hold
at least 1/32 of the size of the database.
==> /fnsw/local/tmp/ora_sig
```

Entering **2** begins the definition of datasets in a remote domain.

```
===== Define FileNet Dataset (Menu 1.1) =====

  [1] - Define datasets for local default domain
  [2] - Define datasets for remote domain
  [3] - Browse all selected domains
  [9] - Help
  [0] - Done
  Successfully defined datasets for local domain

  Enter a command ==> 2
```

At the next prompt, enter the host name of the remote domain on which the dataset you want to define resides (in this example, the host name is corona):

```
Enter a host name that is a server in the remote domain
==>corona

Enter the full path name of Oracle signature file
directory on host 'corona'.
The signature directory should have enough disk space to hold
at least 1/32 of the size of the database.
==> /fnsw/local/tmp/ora_sig

===== Define FileNet Dataset (Menu 1.1) =====

  [1] - Define datasets for local default domain
  [2] - Define datasets for remote domain
  [3] - Browse all selected domains
  [9] - Help
  [0] - Done
Successfully defined datasets in domain contains server
corona
  Enter a command ==> 0
```

Your response to the prompt for the Oracle signature file location completes the FileNet dataset definition for the remote domain. The “Define FileNet Dataset” menu redisplays with a highlighted confirmation message. You can browse your selections, continue with other dataset definitions if necessary, or enter **0** to complete this task.

Tip If you enter **Done** and then **Exit** to exit, EBR_genscript completes the .ddf file creation. You cannot run EBR_genscript again to append additional datasets to an existing .ddf file. Each time you start EBR_genscript, the existing .ddf file is either overwritten or a new .ddf is created. To add to or change an existing .ddf file, you must manually edit the file with a text editor.

You can optionally define generic datasets, including partitions. Enter **2** at the prompt to start generic dataset definition, then enter the host name on which the dataset resides. In this example, a site-controlled Oracle dataset on the host named “venus” is being defined:

```
=====  Define Dataset (Menu 1.0) =====

  [1] - FileNet datasets (required)
  [2] - Generic datasets (optional)
  [9] - Help
  [0] - Done

  Enter a command ==> 2

Enter host ==> venus

(continued on next page)
```

```
(continued from previous page)

===Define Generic Dataset for Host 'venus' (Menu 1.2)==

  [1] - Define partition dataset
  [2] - Define Oracle database dataset

  [9] - Help
  [0] - Done

  Enter a command ==>
```

Defining Oracle Datasets

Enter **2** to define Oracle datasets and respond to the prompts. In this example, the Oracle dataset size is unknown so a carriage return is used to respond to the prompt for dataset size:

```
===Define Generic Dataset for Host 'venus'(Menu 1.2)===

  [1] - Define partition dataset
  [2] - Define Oracle database dataset

  [9] - Help
  [0] - Done

  Enter a command ==> 2

Enter Oracle dataset name ==> oracle_db0

You may provide the dataset size now.  The dataset size is
used for estimating required tape capacity.  To estimate the
dataset size, add the size of all Oracle data files, online
redo logs, and rollback segments.  If you don't have the
information, enter <CR>.
Enter the dataset size in bytes
==> <cr>
```

```
Enter the full path name of Oracle signature file directory
==> /fnsw/local/tmp/ora_sig

Enter the full path name of the Oracle parameter file.
Parameter files are typically named init.ora or init[db_
name].ora
==> /fnsw/local/tmp/init.ora

===Define Generic Dataset for Host 'venus'(Menu 1.2)===

  [1] - Define partition dataset
  [2] - Define Oracle database dataset

  [9] - Help
  [0] - Done
  Successfully defined Oracle dataset on host 'venus'

  Enter a command ==>
```

Tip In the example above, the dataset size is unknown so a carriage return is the response to “Enter dataset size in bytes.” EBR_genscript uses dataset size to estimate the size of the backup media you will need to hold all your data. It is not a required response. If you enter a value in response to this prompt, the value displays as a comment line in the generated script.

Defining Partitions

As previously described, you can define a partition. In this example, the partition resides on the host named “venus.”

From Menu 1.2, “Define Generic Dataset for Host,” enter option **1** and respond to the prompts.

```
===Define Generic Dataset for Host 'venus'(Menu 1.2)===

  [1] - Define partition dataset
  [2] - Define Oracle database dataset

  [9] - Help
  [0] - Done

  Enter a command ==> 1

Enter partition dataset name ==> part

Enter the full path name for the partition filename
Example : /dev/hd2
==> /dev/vg00/fnsw
```

Continue to respond to prompts by entering block size and partition size. When you have completed the dataset definition, enter **0** (Done) to exit this section. Continue to exit until you reach the EBR_genscript main menu.

```
Enter block size in bytes ==>   1024
Enter partition size in blocks, or 0 for the entire partition
==> 0

If you know the partition size, enter partition size now. The
partition size will help to estimate tape capacity. If you
don't know the partition size, enter <CR>.
Enter partition size in bytes ==> <CR>

(continued on next page)
```

```
(continued from previous page)

===Define Generic Dataset for Host 'venus'(Menu 1.2)===

  [1] - Define partition dataset
  [2] - Define Oracle database dataset

  [9] - Help
  [0] - Done
 Successfully defined partition dataset on host 'venus'

  Enter a command ==> 0
```

At the main menu, you can enter options to generate a device specification file and to create a backup or restore script or enter **0** to exit EBR_genscript.

Below is the .ddf generated by the multiple domain example on the previous pages. Your .ddf file will be different and may or may not include remote domains or generic (non-FileNet datasets), as this example does:

```
DATASETS

------------------- Domain             Mars:FileNet -------------------
--  Mars             HP9000 : Server ID: 1, Server Type : Combined
-----------------------------------------------------------------------

-- Begin Datasets for Server 'Mars' of 'Mars:FileNet' -----------------

  Oracle_Mars : Oracle
--  DatasetSize = 314,572,800
    location = "Mars";
    signature_file_directory = "/fnsw/local/tmp/ora_sig";
  end_Oracle
```

```
  PermDB_Mars : MKF
--  DatasetSize = 146,800,640
    location = "Mars";
    base_data_file = "/fnsw/dev/1/permanent_db0";
  end_MKF

  SecDB_Mars : MKF
--  DatasetSize = 16,777,216
    location = "Mars";
    base_data_file = "/fnsw/dev/1/sec_db0";
  end_MKF

  TranDB_Mars : MKF
--  DatasetSize = 62,914,560
    location = "Mars";
    base_data_file = "/fnsw/dev/1/transient_db0";
    transient_db;
  end_MKF

  PageCache_Mars : cache
--  DatasetSize = 83,886,080
    location = "Mars";
    transient_db = TranDB_Mars;
    permanent_db = PermDB_Mars;
    security_db = SecDB_Mars;
  end_cache

-- End Datasets for Server 'Mars' of 'Mars:FileNet' -------------------

------------------- Domain           corona:FileNet -------------------
--  corona           AIX    : Server ID: 1, Server Type : Combined
-----------------------------------------------------------------------

-- Begin Datasets for Server 'corona' of 'corona:FileNet' -----------------

  Oracle_corona : Oracle
--  DatasetSize = 314,572,800
    location = "corona";
    signature_file_directory = "/fnsw/local/tmp/ora_sig";
```

```
  end_Oracle

  PermDB_corona : MKF
--  DatasetSize = 146,800,640
    location = "corona";
    base_data_file = "/fnsw/dev/1/permanent_db0";
  end_MKF

  SecDB_corona : MKF
--  DatasetSize = 16,777,216
    location = "corona";
    base_data_file = "/fnsw/dev/1/sec_db0";
  end_MKF

  TranDB_corona : MKF
--  DatasetSize = 62,914,560
    location = "corona";
    base_data_file = "/fnsw/dev/1/transient_db0";
    transient_db;
  end_MKF
  PageCache_corona : cache
--  DatasetSize = 83,886,080
    location = "corona";
    transient_db = TranDB_corona;
    permanent_db = PermDB_corona;
    security_db = SecDB_corona;
  end_cache

-- End Datasets for Server 'corona' of 'corona:FileNet' -------------------

-- Begin Generic Datasets for host 'venus' -------------------

  oracle_db0 : Oracle
    location = "venus";
    signature_file_directory = "/fnsw/local/tmp/ora_sig";
    parameter_file = "/fnsw/local/tmp/init.ora";
  end_Oracle

  part : partition
    location = "venus";
```

```
    filename = "/dev/vg00/fnsw";
  end_partition

-- End Generic Datasets for host 'venus' -------------------

-------------------- Order Constraints --------------------

  order_constraints
    SecDB_Mars, PermDB_Mars, TranDB_Mars before PageCache_Mars;
    SecDB_corona, PermDB_corona, TranDB_corona before PageCache_corona;
  end_order_constraints

END_DATASETS

-----------------------------------------------------------------
-- This file contains datasets for the following domains:
--
--  Mars:FileNet
--  corona:FileNet
--  venus:FileNet
--
-----------------------------------------------------------------
```

Creating a Device Specification File

The examples in this section define a tape library and a tape as the device type for the backup. You may define the device as a disk file, a tape in a standalone tape drive, or a tape in a tape library.

The first example defines a tape library. From the main menu, enter **2** and respond to the prompts.

```
== EBR_genscript Main Menu

  [1] - Generate dataset definitions
  [2] - Generate device specification
  [3] - Generate backup script
  [4] - Generate restore script

  [9] - Help
  [0] - Exit

  Enter a command ==> 2

Enter EBR device specification name
(suffix '.dev' will be appended to the file name entered)

==> mars
```

From the Define Device Specification Menu, enter 3 and respond to the prompts.

```
===== Define Device Specification (Menu 2.0) =====
  [1] - Disk file
  [2] - Tape
  [3] - Tape library

  [9] - Help
  [0] - Done

  Enter a command ==> 3

Enter tape library device id (default id LIB1) ==> <CR>
Enter the device location (<CR> = local host) ==> <CR>

(continued on next page)
```

```
(continued from previous page)

Enter the full path name of tape library device file ==> /dev/
chgr0
Select tape library type:
       [1] EXB-210     [2] EXB-218     [3] EXB-220
==> 2
Number of tape drives ==> 1
Enter the full path name of rewind device name (drive 1)
==> /dev/rmt/0m
Enter the full path name of no-rewind device name (drive 1)
==> /dev/rmt/0mn
Enter a short description with less than 24 characters, or
enter <CR> to continue ==> <CR>
```

The second example defines a tape as the device for the backup. From the main menu, enter **2** and respond to the prompts.

```
== EBR_genscript Main Menu

  [1] - Generate dataset definitions
  [2] - Generate device specification
  [3] - Generate backup script
  [4] - Generate restore script

  [9] - Help
  [0] - Exit

  Enter a command ==> 2

Enter EBR device specification name
(suffix '.dev' will be appended to the file name entered)

==> mars
```

From the Define Device Specification Menu, enter 2 and respond to the prompts.

```
===== Define Device Specification (Menu 2.0) =====
  [1] - Disk file
  [2] - Tape
  [3] - Tape library

  [9] - Help
  [0] - Done

  Enter a command ==> 2

Enter tape device id (default id TAPE_DEV1) ==> <CR>
Enter the device location (<CR> = local host) ==> <CR>
Enter the full path name of rewind device name ==>      /dev/
rmt/0m
Enter the full path name of no-rewind device name ==>   /dev/
rmt/0mn
Select tape device type: [1] 8mm,   [2] 4mm or DAT,   [3] QIC
==> 2
Enter a short description with less than 24 characters, or
enter <CR> to continue ==> <CR>
```

The device specifications file output generated from the two examples on the previous pages is displayed below:

```
DEVICE_SPECIFICATIONS

  LIB1 : tape_library
    location = "mars";
    library_device = "/dev/chgr0";
    library_type = "EXB-218";

    tape_drives
      num_drives = 1;
      drive 1
```

```
        drive_name_rewind = "/dev/rmt/0m";
        drive_name_no_rewind = "/dev/rmt/0mn";
        drive_type = 4mm;

    end_tape_drives

  end_tape_library

  TAPE_DEV1 : tape_drive
    location = "mars";
    drive_name_rewind = "/dev/rmt/0m";
    drive_name_no_rewind = "/dev/rmt/0mn";
    drive_type = 4mm;
  end_tape_drive

END_DEVICE_SPECIFICATIONS
```

Creating a Backup Script

The example in this section builds a backup script using a predefined dataset definitions file (see **"Creating a Dataset Definition File" on page 279**).

To build a restore script (suffix .res), enter **4** and refer to **"Creating a Restore Script" on page 319**.

Note The backup script generated by the responses in this example are included under **"Backup Script Output" on page 316**. Your backup script will be different.

From the main menu, enter **3** to create a backup script and respond to the prompts:

```
== EBR_genscript Main Menu

  [1] - Generate dataset definitions
  [2] - Generate device specification
  [3] - Generate backup script
  [4] - Generate restore script
  [9] - Help
  [0] - Exit

  Enter a command ==> 3

Enter EBR backup script name
(suffix '.bac' will be appended to the file name entered)
 ==> /fnsw/local/EBR/tt

Backup Global Parameters Section:
The backup global parameters specify the volume group name and
backup media expiration limit.

Specify the volume group name ==> TEST_VG

Specify the backup media expiration in days ==> 30

Dataset Definition File:
Enter dataset definition file directory (<CR> = current
directory) ==> /fnsw/local/EBR

(continued on next page)
```

```
(continued from previous page)

== Dataset definition file list ==
/fnsw/local/tmp/EBR/tt.ddf

Enter the dataset definition file ==>
/fnsw/local/EBR/tt.ddf

Device Specification File:
Enter the directory path which contains device specification
files (<CR> = current directory)==> <CR>

== Device Specification file list ==
   /fnsw/local/EBR/gen/mars.dev

Enter the device specification file ==> mars.dev
```

Note If you have created more than one .ddf file in the same directory, EBR_genscript displays a list of all .ddf files and prompts you to enter the one you want to use for this script. The full path name is required only if the .ddf file is not in the current directory.

The next menu guides you through dataset selection:

```
=== Select Dataset for Backup (Menu 3.0) Operation ===

  [1] -  Select datasets from dataset list
  [2] -  List selected datasets
  [3] -  Browse pre-defined datasets

  [9] -  Help
  [0] -  Done

  Enter a command ==>
```

Before you make your selections, you can use “Browse pre-defined datasets” to display all the datasets contained in the .ddf file.

Tip The Browse option displays a list of all the datasets in the .ddf file. You cannot select datasets from this list. You must use the ”Select datasets from dataset list” option to choose datasets for inclusion in your script.

You can select certain datasets in the .ddf file for inclusion in your script or you can select all datasets. After you select datasets, you can review your choices with the “List selected datasets” option.

In the following sequence, all datasets are selected for inclusion in the script:

Tip If you responded with a carriage return to prompts for dataset or partition size as you created your dataset definition file, the “Size” column reflects an “unknown” size.

```
===== Select Dataset for Backup (Menu 3.0) Operation =====

  [1] -  Select datasets from dataset list
  [2] -  List selected datasets
  [3] -  Browse pre-defined datasets
  [9] -  Help
  [0] -  Done

  Enter a command ==> 1

(continued on next page)
```

```
(continued from previous page)

    Dataset Name                Location              Size
                                                   (uncompressed)

  1: Oracle_Mars                "Mars"              314,572,800
  2: PermDB_Mars                "Mars"              146,800,640
  3: SecDB_Mars                 "Mars"               16,777,216
  4: TranDB_Mars                "Mars"               62,914,560
  5: PageCache_Mars             "Mars"               83,886,080
  6: Oracle_corona              "corona"            314,572,800
  7: PermDB_corona              "corona"            146,800,640
  8: SecDB_corona               "corona"             16,777,216
  9: TranDB_corona              "corona"             62,914,560
 10: PageCache_corona           "corona"             83,886,080
 11: oracle_db0                 "venus"                 UNKNOWN
 12: part                       "venus"                 UNKNOWN

Enter dataset numbers separated by spaces, or type 'ALL' to
select all datasets, or type <CR> to quit selection.
==> all
```

When you complete your selections, the menu displays followed by a completion message indicating the number of datasets selected:

```
=== Select Dataset for Backup (Menu 3.0) Operation ===

  [1] -  Select datasets from dataset list
  [2] -  List selected datasets
  [3] -  Browse pre-defined datasets

  [9] -  Help
  [0] -  Done
  12 datasets are selected
```

You can review these selections with the List (**l**) option or if you are satisfied with the selections, enter **0** to continue to the next task.

When you enter **0**, additional prompts guide you through the process of establishing the backup environment for your script. You must reply to:

- the number of threads to be used in your backup
- the backup mode (offline or online)
- the archive log retention period
- the backup type (full or interval)
- the backup environment such as the name of the host providing tape service, volume serial number, backup media type, backup device type (if you are using tape)
- other device-related information

The example below is for illustrative purposes and not all datasets are shown):

```
 Enter a command ==> 0

To estimate the number of threads to be used for backup, you
may enter the estimated backup media capacity in megabytes, or
enter 0 to skip
the estimation process.
==> 2000

You have selected the media capacity = 2,097,152,000 (2000
Mb).  20% is reserved, 80% available (1,677,721,600 bytes).
Total compressed dataset size = 660,602,880 bytes exclude 2
datasets with unknown dataset size. Cache datasets are non-
compressible.  Cache datasets size is estimated based on page
cache and batch cache as 100% full with locked objects.  The
estimated cache size can be reduced based on cache utilization
which can be obtained from CSM_tool statistics report.

(continued on next page)
```

```
(continued from previous page)

Suggestion: use 1 thread  to backup.
Specify the number of threads to be used for backup operation
==> 1
Select backup mode for Host: "Mars"
Online backup [1] or Offline backup [2]
==> 2

===== Select Backup Option for 'Oracle_Mars' =====
     Host: "Mars"               Dataset type : Oracle Database

Full backup [1] or Interval backup [2]
==> 1
Enter archive log retention days (minimum 2 days)
==> 14
===== Select Backup Option for 'PermDB_Mars' =====
     Host: "Mars"                Dataset type : MKF

Full backup [1] or Interval backup [2]
==> 1
===== Select Backup Option for 'SecDB_Mars' =====
     Host: "Mars"                Dataset type : MKF

Full backup [1] or Interval backup [2]
==> 1
...
Specify volume serial number for thread 1
==> TAPE_SER1
```

At the point in the example above where you are asked to enter the volume serial number, you need a different entry if your backup device is a tape library with a barcode reader and you use tapes with barcode labels. You need to enter the volume serial number in the format, BAR_ascii_characters, to represent the tape serial number. For example, enter BAR_0002000 or BAR_CNL001, where the ascii_characters part of the serial number must be the same as the bar code label. See the

description for tape serial number in **"EBR_label" on page 332** for more details.

Your responses to the prompts in the example above can be changed by editing the output file so do not be concerned with overestimating or underestimating such things as tape capacity or number of threads.

After specifying the volume serial number for thread 1, the next screen, Select Device for Thread 1 Backup, displays.

```
=== Select Device for Thread 1 Backup (Menu 3.1) ===

  [1] -  Select device from device list
  [2] -  Browse pre-defined device

  [9] -  Help
  [0] -  Done
Enter a command ==> 1

 Device ID   Location  Type     Description
 ==========   =========   ======    =============
1: LIB1       mars        Library   /dev/chgr0(EXB-218)
2: TAPE_DEV1  mars        Tape      /dev/rmt/0m (4mm)

Enter device id numbers ==> 1
Thread 1: Select tape drive (id 1-2  ==> 1
Thread 1: To load tape in tape library, tape must be specified
by either volume serial number which must be in the format of
'BAR_' followed by an ASCII string, or specified by a slot
number. Since the Volume Serial Number specified in thread 1
is 'TAPE_SER1' which is not in bar code format, slot id is
required. Select slot id (1-18)
==> 1
LIB1(drive 1, slot 1) is selected for thread 1. Is this OK [y/
n] ==> y
```

When you have responded to the last prompt, the “Select Dataset for Thread *n* (Menu 3.2)” displays. If you defined multiple threads, this menu displays sequentially for each thread. You can assign datasets to different threads or all datasets to a single thread, depending on your backup strategy. Generally, multiple threads imply use of multiple tape drives for the backup.

Note EBR does not support the Exabyte tape library on the Solaris and Windows Server platforms.

In the following example, all datasets are assigned to a single thread:

```
===== Select Dataset for Thread 1 (Menu 3.2) =====

  [1] -  Assign datasets to thread 1
  [2] -  Review datasets assigned to thread 1
  [3] -  List all pre-selected datasets for Backup operation

  [9] -  Help
  [0] -  Done

  Enter a command ==> 1

(continued on next page)
```

```
(continued from previous page)

    Dataset Name      Location  Size  Stripe
                                uncompressed  defined/selected

 1: Oracle_Mars       "Mars"    314,572,800    1/0
 2: PermDB_Mars       "Mars"    146,800,640    1/0
 3: SecDB_Mars        "Mars"    16,777,216     1/0
 4: TranDB_Mars       "Mars"    62,914,560     1/0
 5: PageCache_Mars    "Mars"    83,886,080     1/0
 6: Oracle_corona     "corona"  314,572,800    1/0
 7: PermDB_corona     "corona"  146,800,640    1/0
 8: SecDB_corona      "corona"  16,777,216     1/0
 9: TranDB_corona     "corona"  62,914,560     1/0
10: PageCache_corona  "corona"  83,886,080     1/0
11: oracle_db0        "venus"   UNKNOWN        1/0
12: part              "venus"   UNKNOWN        1/0

Thread 1: Dataset list is sorted by order constraint. Select
datasets in numeric ascending order separated by spaces, or
type 'ALL' to select all, or type <CR> to quit selection.
==> all
```

When dataset-to-thread assignment is complete, Menu 3.2 redisplays with a confirmation message:

```
===== Select Dataset for Thread 1 (Menu 3.2) =====

  [1] -  Assign datasets to thread 1
  [2] -  Review datasets assigned to thread 1
  [3] -  List all pre-selected datasets for Backup
         operation

  [9] -  Help
  [0] -  Done
  Thread 1: assigned 12 datasets

  Enter a command ==>
```

Use the **2** option to review your assignments. If you are satisfied with the assignments, enter **0** to exit this menu. If you do not want to generate your restore script at this time, enter **e** to exit the EBR_genscript tool.

Backup Script Output

Use a viewing command or text editor to examine the output (.bac) file. The output file is in ASCII format that you can edit with a text editor.

The /fnsw/local/tmp/EBR/tt.bac file below is the backup script generated by the example on the previous pages:

```
Mars(user1)/fnsw/local> more /fnsw/local/tmp/EBR/tt.bac

EBR_script ( format_level = 2; );

BACKUP_GLOBAL_PARAMETERS
  volume_group = TEST_VG;
  expiration = 30 days;
END_BACKUP_GLOBAL_PARAMETERS

#include "/fnsw/local/EBR/tt.ddf"

#include "/fnsw/local/EBR/gen/mars.dev"

BACKUP_OPTIONS
   Oracle_Mars : backup_options
    full_backup;
    offline_backup;
    archive_redo_log_retention = 14 day;
  end_backup_options
```

```
PermDB_Mars : backup_options
  full_backup;
  offline_backup;
end_backup_options

SecDB_Mars : backup_options
  full_backup;
  offline_backup;
end_backup_options

TranDB_Mars : backup_options
  full_backup;
  offline_backup;
end_backup_options

PageCache_Mars : backup_options
  full_backup;
end_backup_options

Oracle_corona : backup_options
  full_backup;
  offline_backup;
  archive_redo_log_retention = 14 day;
end_backup_options

PermDB_corona : backup_options
  full_backup;
  offline_backup;
end_backup_options

SecDB_corona : backup_options
  full_backup;
  offline_backup;
end_backup_options
```

```
  TranDB_corona : backup_options
    full_backup;
    offline_backup;
  end_backup_options

  PageCache_corona : backup_options
    full_backup;
  end_backup_options

  oracle_db0 : backup_options
    full_backup;
    offline_backup;
    archive_redo_log_retention = 14 day;
  end_backup_options

END_BACKUP_OPTIONS

THREADS

  num_threads = 1;
  thread 1
  device = LIB1(drive 1, slot 1);
  volume_serial = TAPE_SER1;
  datasets
     Oracle_Mars          (part 1 of 1),
     PermDB_Mars          (part 1 of 1),
     SecDB_Mars           (part 1 of 1),
     TranDB_Mars          (part 1 of 1),
     PageCache_Mars       (part 1 of 1),
     Oracle_corona        (part 1 of 1),
     PermDB_corona        (part 1 of 1),
     SecDB_corona         (part 1 of 1),
     TranDB_corona        (part 1 of 1),
```

```
    PageCache_corona      (part 1 of 1),
    oracle_db0            (part 1 of 1),
    part                  (part 1 of 1);
  end_thread      -- thread 1 --
END_THREADS
```

Creating a Restore Script

Building a restore script is similar to building a backup script. (To build a backup script, refer to **"Creating a Backup Script" on page 305**.)

Note The restore script generated by the responses in this example are included under **"Restore Script Output" on page 325**. Your restore script will be different.

To build a restore script (suffix .res), enter **4** at the EBR_genscript Main Menu and respond to the prompts:

```
== EBR_genscript Main Menu

  [1] - Generate dataset definitions
  [2] - Generate device specification
  [3] - Generate backup script
  [4] - Generate restore script

  [9] - Exit
  [0] - Help

  Enter a command ==> 4

(continued on next page)
```

```
(continued from previous page)

Enter EBR restore script name
(suffix '.res' will be appended to the file name entered)

==> /fnsw/local/EBR/tt

Restore Global Parameters Section:
The restore global parameters specify the volume group name.
Specify the volume group name
==> TEST_VG

Dataset Definition File
Enter dataset definition file directory
(<CR> = current directory)==> /fnsw/local/EBR

== Dataset definition file list ==
/fnsw/local/tmp/EBR/tt.ddf

Enter the dataset definition file
==> /fnsw/local/EBR/tt.ddf
```

You are then prompted for the device specification file and datasets for restore as illustrated below:

```
Device Specification File:
Enter the directory path which contains device specification
files (<CR> = current directory)==> <CR>

(continued on next page)
```

```
(continued from previous page)

== Device Specification file list ==
/fnsw/local/tmp/EBR/gen/mars.dev

Enter the device specification file
==> mars.dev

=== Select Dataset for Restore (Menu 4.0) Operation ===

  [1] -  Select datasets from dataset list
  [2] -  List selected datasets
  [3] -  Browse pre-defined datasets

  [9] -  Help
  [0] -  Done

  Enter a command ==>
```

You can browse the list of datasets in the .ddf file or enter **1** to begin choosing datasets to include in your restore script. In the example below, two datasets are selected from the list and the result is examined with the “List selected datasets” command:

Tip If you responded with a carriage return to prompts for dataset or partition size as you created your dataset definition file, the “Size” column of the dataset list reflects an “unknown” size.

```
 Enter a command ==> 1

(continued on next page)
```

```
(contined from previous page)

    Dataset Name                Location                    Size
                                                       (uncompressed)

  1: Oracle_Mars                "Mars"                 314,572,800
  2: PermDB_Mars                "Mars"                 146,800,640
  3: SecDB_Mars                 "Mars"                  16,777,216
  4: TranDB_Mars                "Mars"                  62,914,560
  5: PageCache_Mars             "Mars"                  83,886,080
  6: Oracle_corona              "corona"               314,572,800
  7: PermDB_corona              "corona"               146,800,640
  8: SecDB_corona               "corona"                16,777,216
  9: TranDB_corona              "corona"                62,914,560
 10: PageCache_corona           "corona"                83,886,080
 11: oracle_db0                "venus"                    UNKNOWN
 12: part                      "venus"                    UNKNOWN

Enter dataset numbers separated by spaces, or type 'ALL' to
select all datasets.
==> 4 5

===== Select Dataset for Restore (Menu 4.0) Operation =====

  [1] -  Select datasets from dataset list
  [2] -  List selected datasets
  [3] -  Browse pre-defined datasets

  [9] -  Help
  [0] -  Done
  2 datasets are selected

  Enter a command ==> 2

    Dataset Name              Location                  Size
                                                  (uncompressed)

 1: TranDB_Mars               "Mars"                  62,914,560
 2: PageCache_Mars            "Mars"                  83,886,080
-- End -- (<CR> to continue)
```

When you are satisfied with your selections, enter **0** to exit the menu and continue with thread assignment:

```
Enter a command ==> 0

Specify the number of threads to be used for restore operation
==> 1

Thread 1 Options:
Specify volume serial number for thread 1 ==> TAPE_SER1

=== Select Device for Thread 1 Restore (Menu 4.1) ===

  [1] -  Select device from device list
  [2] -  Browse pre-defined device
  [9] -  Help
  [0] -  Done
Enter a command ==> 1

 Device ID  Location   Type      Description
 ==========   =========   ====       =============
1: LIB1       mars        Library    /dev/chgr0(EXB-218)
2: TAPE_DEV1  mars        Tape       /dev/rmt/0m (4mm)

Enter device id numbers ==> 1
```

Because the device selected is a tape library, the following prompts request the tape drive id and whether the tape is specified by volume serial number or slot number.

```
Thread 1: Select tape drive (id 1-2)
==> 1
Thread 1: To load tape in tape library, tape must be specified
by either volume serial number which must be in the format of
'BAR_' followed by an ASCII string, or specified by a slot
number. Since the Volume Serial Number specified in thread 1
is 'TAPE_SER1' which is not in bar code format, slot id is
required. Select slot id (1-18)
==> 1
LIB1(drive 1, slot1) is selected for thread 1. Is this OK [y/
n] ==> y
```

After selecting the device for the restore, the next screen displays:

```
===== Select Dataset for Thread 1 (Menu 4.2) =====

  [1] -  Assign datasets to thread 1
  [2] -  Review datasets assigned to thread 1
  [3] -  List all pre-selected datasets for Restore
         operation

  [9] -  Help
  [0] -  Done

  Enter a command ==> 1

    Dataset Name   Location  Size          Stripe
                             uncompressed  defined/selected
 1: TranDB_Mars    "Mars"    62,914,560     1/0
 2: PageCache_Mars "Mars"    83,886,080     1/0

(continued on next page)
```

```
(continued from previous page)

===== Select Dataset for Thread 1 (Menu 4.2) =====

  [1] -  Assign datasets to thread 1
  [2] -  Review datasets assigned to thread 1
  [3] -  List all pre-selected datasets for Restore
         operation

  [9] -  Help
  [0] -  Done
  Thread 1: assigned 2 datasets

 Enter a command ==>
```

When you are satisfied with your script, enter **0** to exit the EBR_genscript program. Examine the following output (.res) file.

Restore Script Output

The /fnsw/local/tmp/EBR/tt.res file below was generated by the example on the previous pages:

```
Mars(user1)/fnsw/local> more /fnsw/local/EBR/tt.res

EBR_script ( format_level = 2; );

RESTORE_GLOBAL_PARAMETERS
  volume_group = TEST_VG;
END_RESTORE_GLOBAL_PARAMETERS

#include "/fnsw/local/EBR/tt.ddf"

#include "/fnsw/local/EBR/gen/mars.dev"

RESTORE_OPTIONS

  TranDB_Mars : restore_options
```

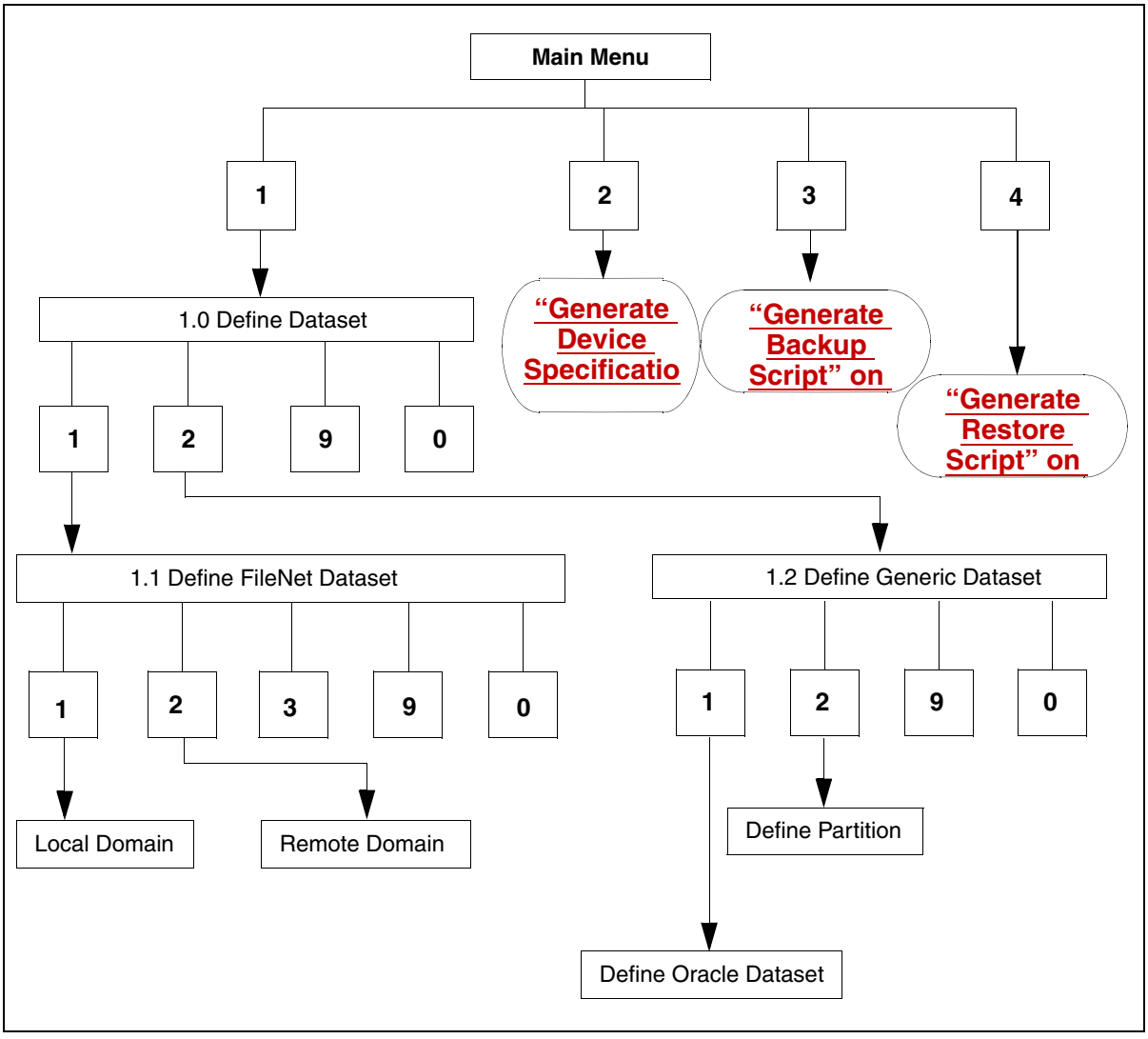
```
    full_restore;
      interval_restore_follows = false;
  end_restore_options

  PageCache_Mars : restore_options
    full_restore;
      interval_restore_follows = false;
  end_restore_options

END_RESTORE_OPTIONS

THREADS

  num_threads = 1;
  thread 1
    device = LIB1 (drive 1, slot 1);
    volume_serial = TAPE_SER1;
    datasets
      TranDB_Mars          (part 1 of 1),
      PageCache_Mars       (part 1 of 1);
  end_thread     -- thread 1 --

END_THREADS
```

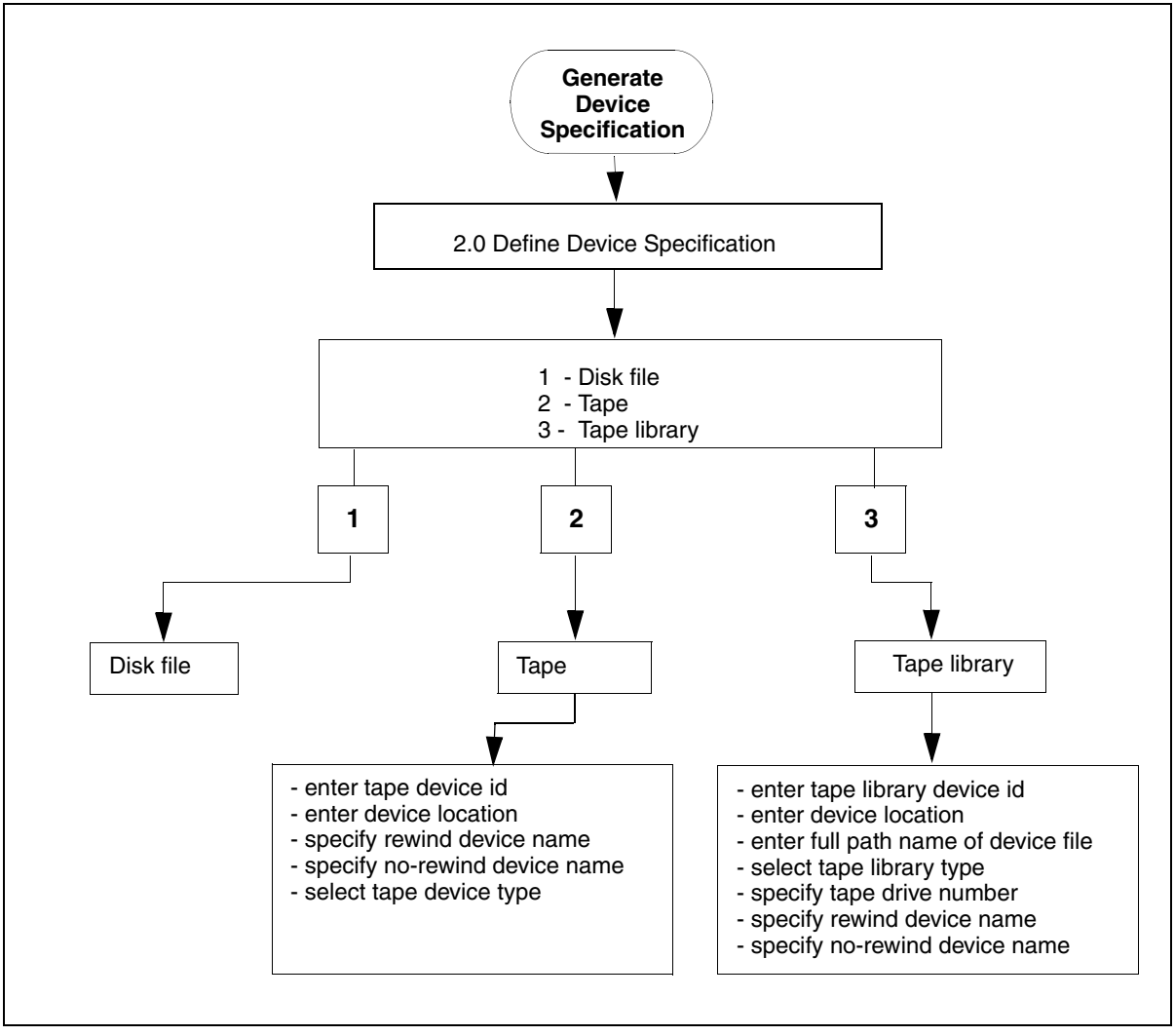
EBR_genscript Flow Chart

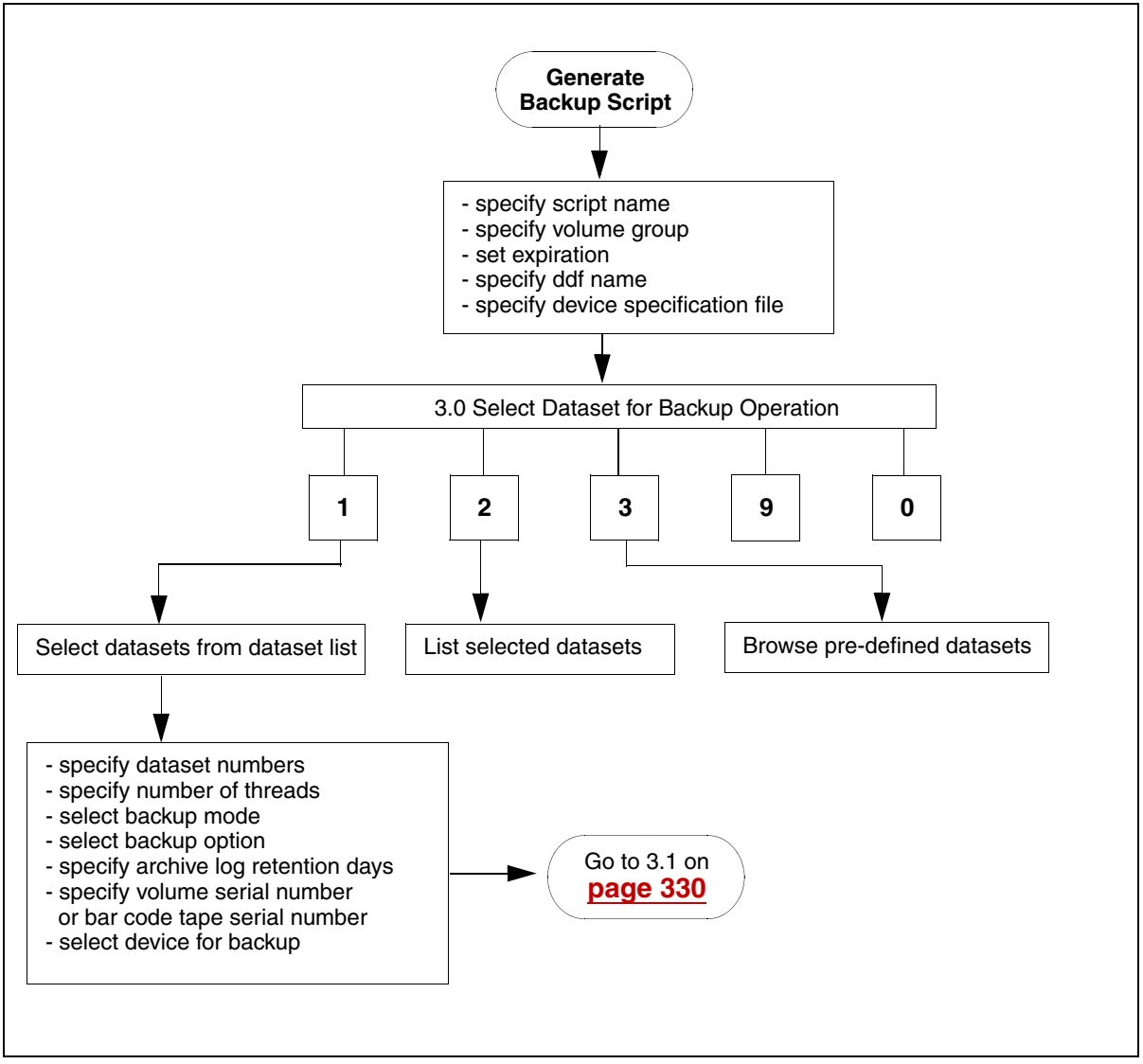
The flowchart in this section shows the paths you can take through EBR_genscript to create data definition files and scripts.

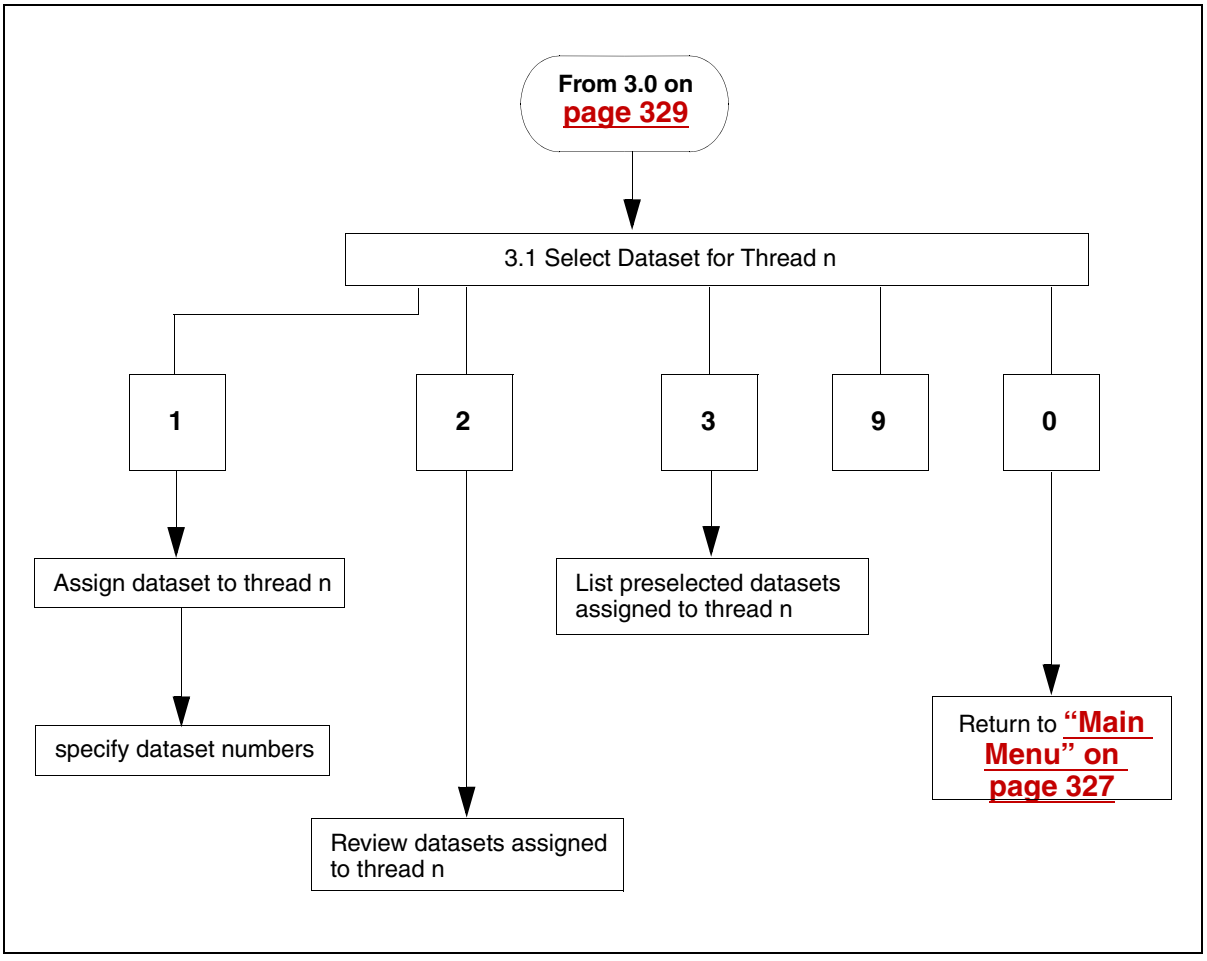
Major menu headings are numbered. In each menu, entering **9** displays help and **0** ends the task and continues to the next task in the sequence. Since the **9** and **0** options are the same for every menu, they are illustrated in detail in the flowchart only for special situations.

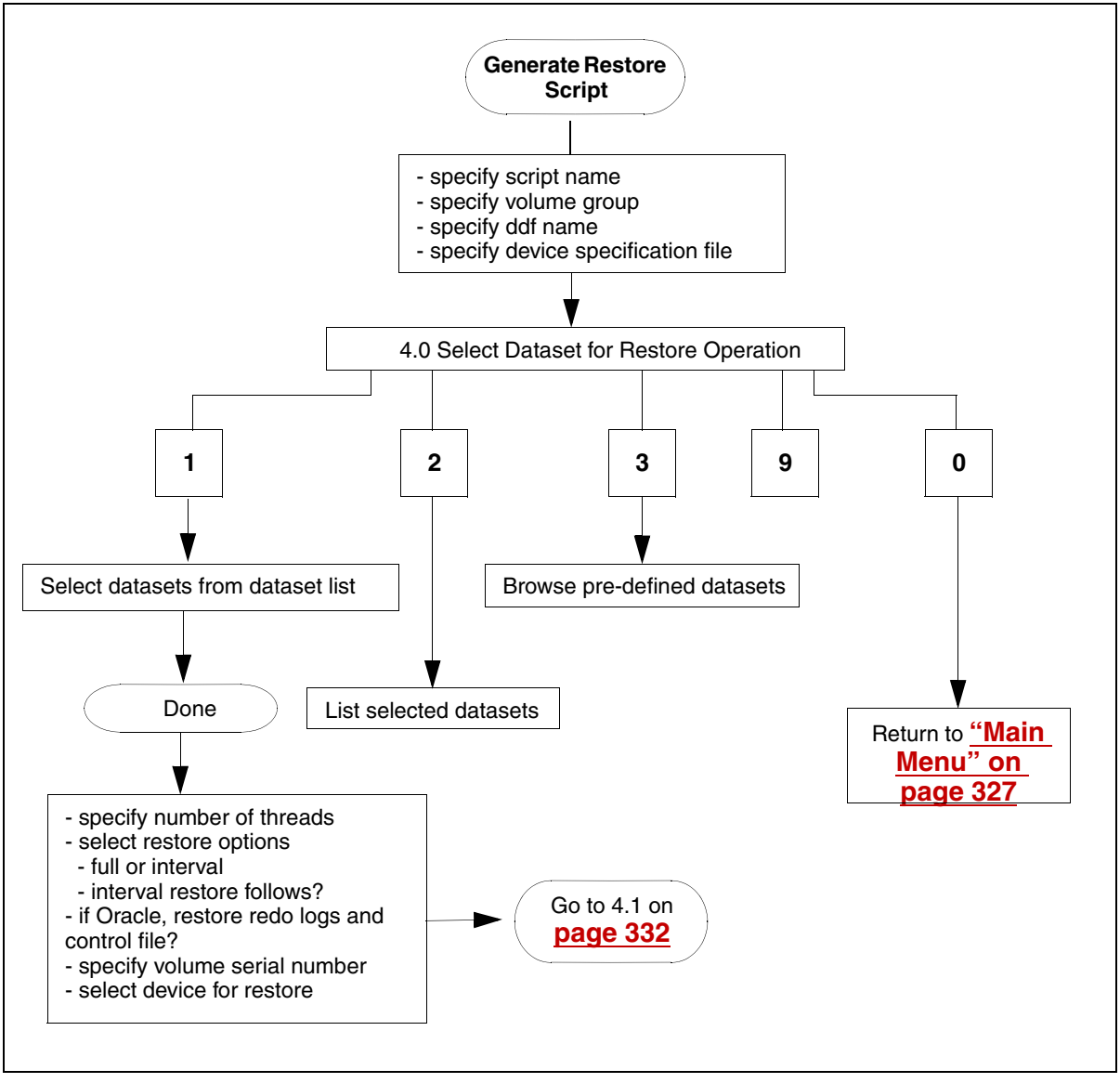
Note Not every option is described in the flowchart. Some options prompt you for additional specifications. Such specifications are listed under the option in the flowchart.

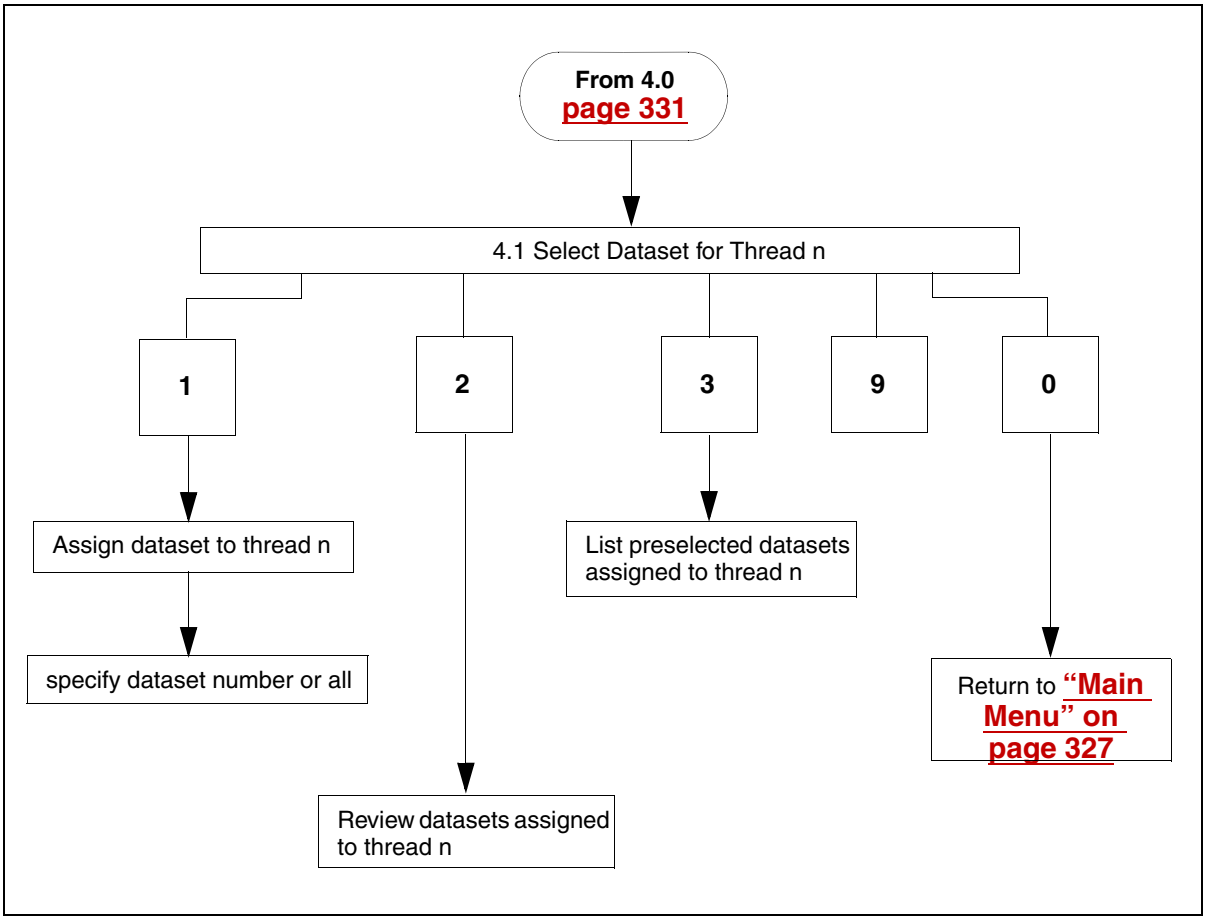

Main Menu
1
2
3
4
1.0 Define Dataset
"Generate Device Specificatio
"Generate Backup Script" on
"Generate Restore Script" on
1
2
9
0
1.1 Define FileNet Dataset
1.2 Define Generic Dataset
1
2
3
9
0
1
2
9
0
Local Domain
Remote Domain
Define Partition
Define Oracle Dataset


Generate Device Specification
2.0 Define Device Specification
1 - Disk file
2 - Tape
3 - Tape library
1
2
3
Disk file
Tape
Tape library
- enter tape device id
- enter device location
- specify rewind device name
- specify no-rewind device name
- select tape device type
- enter tape library device id
- enter device location
- enter full path name of device file
- select tape library type
- specify tape drive number
- specify rewind device name
- specify no-rewind device name


Generate Backup Script
- specify script name
- specify volume group
- set expiration
- specify ddf name
- specify device specification file
3.0 Select Dataset for Backup Operation
1
2
3
9
0
Select datasets from dataset list
List selected datasets
Browse pre-defined datasets
- specify dataset numbers
- specify number of threads
- select backup mode
- select backup option
- specify archive log retention days
- specify volume serial number
or bar code tape serial number
- select device for backup
Go to 3.1 on
page 330


From 3.0 on
page 329
3.1 Select Dataset for Thread n
1
2
3
9
0
Assign dataset to thread n
specify dataset numbers
List preselected datasets assigned to thread n
Review datasets assigned to thread n
Return to "Main Menu" on page 327


Generate Restore Script
- specify script name
- specify volume group
- specify ddf name
- specify device specification file
4.0 Select Dataset for Restore Operation
1
2
3
9
0
Select datasets from dataset list
Browse pre-defined datasets
Done
List selected datasets
Return to "Main Menu" on page 327
- specify number of threads
- select restore options
- full or interval
- interval restore follows?
- if Oracle, restore redo logs and control file?
- specify volume serial number
- select device for restore
Go to 4.1 on page 332



EBR_label

For EBR backups, always use tapes that you have prelabeled with the EBR_label utility.

The EBR_label utility labels and relabels the following:

- tapes in a standalone tape drive used for backup
- tapes in a tape library used for backup
- magnetic disk files used for backup

Note The tape library is not supported on the Solaris and Windows Server platforms.

If your system is UNIX-based, run EBR_label in a shell window. If your operating system is Windows Server Server, run EBR_label in an MS-DOS window.

Tip When using standard tape drives, make sure you insert a tape into the tape drive before you start the EBR_label program. If a tape is not in the drive when EBR_label starts, the program terminates with an error message indicating the tape drive is offline.

Caution A common mistake in the backup process is to insert a tape with the wrong label into the tape drive. This mistake will lock up the datasets. To unlock the datasets, you must use the the following utilities: **"EBR_orreset" on page 344** and **"EBR_ulmk" on page 351**.

The command line parameters to EBR_label are:

- Full path name (disk or tape or tape library)
- Tape device type
- Tape serial number
- Volume group name

- Library device type (not supported on Solaris and Windows Server)
- Slot number (not supported on Solaris and Windows Server)

The command line parameters are in **keyword=value** form with no embedded blanks. If run without parameters or with incorrect parameters, EBR_label displays an extensive help message.

The command line syntax for the initial labeling of a tape or disk file is slightly different than if you are relabeling a tape or disk file. If you are labeling a tape for the first time in its life, see **"Initial Tape Labeling" on page 334**. See **"Relabeling a Disk File or a Tape in a Tape Drive or Tape Library" on page 341** for other parameters if you are relabeling a tape.

Initial Tape Labeling

EBR_label can be used to label a tape in a standalone tape drive or in a tape library, or to label a disk file.

Command line syntax for labeling a tape in a standalone tape drive is:

```
EBR_label tape=<full_path_name> type=<tape_device_type>  \
    ser=<tape_serial_number> vg=<volume_group_name>
```

Command line syntax for labeling a tape in a tape library is:

```
EBR_label lib=<full_path_name> lib_type=<lib_device_type> \
    lib_drive=<drive_num> [slot=<slot_num>] \
    tape=<full_path_name> type=<tape_device_type> \
    ser=<tape_serial_number> vg=<volume_group_name>
```

Command line syntax for labeling of a disk file is:

```
EBR_label disk=<full_path_name> ser=<tape_serial_number>  \
    vg=<volume_group_name>
```

where

<full_path_name> is the full path name of the rewind tape drive or the tape library device or the disk file. (For the rewind tape drive, use the highest density available.)

Note EBR checks both the rewind and non-rewind tape device names and selects the appropriate device. However, the EBR_label and EBR_tdir utilities require that you specify only one tape device name. You can specify either the rewind or no-rewind name.

<tape_device_type> is specified for labeling tape in a standalone tape drive or tape library. It is one of the following keywords:

Keyword Value	Description
8mm	8 millimeter cartridge tape
4mm	4 millimeter cartridge tape
DAT	Digital audio tape, a synonym for 4mm
QIC	Quarter-inch cartridge
DLT	Digital Linear Tape

You must specify the same tape device type in both the EBR_label command and your backup script. Failing to do so may result in an inability to restore from the backup tape. See **"Selecting Tape Device Type" on page 132** for more information.

<tape_serial_number> is a unique serial number consisting of any combination of ASCII upper case letters, ASCII decimal digits, and the

ASCII underscore character. Serial numbers can be from 1 to 12 characters long. The first character must be an ASCII uppercase letter.

For a tape library labeling, if a tape has a bar code label and a bar code reader is present, the tape serial number must be entered as:

BAR_ascii_characters

For example, BAR_0002000 or BAR_CNL001, where the ascii_characters part of the serial number must be the same as the bar code label.

<volume_group_name> is the volume group name consisting of any combination of ASCII upper case letters, ASCII decimal digits, and the ASCII underscore character. A volume group name can be from 1 to 28 characters long and the first character must be an ASCII upper case letter. The volume group name should be unique for each set of backup tapes in a backup cycle.

<lib_device_type> is specified only for labeling tape in a tape library and is one of the following keywords. The keyword may be enclosed in double quotes, except for the Exabyte keyword, which does not require double quotes.

Keyword Value	Description
“EXB-210”	Exabyte 8 millimeter cartridge tape
“EXB-218”	Exabyte 4 millimeter cartridge tape
“EXB-220”	Exabyte 8 millimeter “Mammoth” cartridge tape
Exabyte	Refers to any of the above Exabyte cartridge tapes and can be used if you are not sure of the type

<drive_num> is tape drive location in the tape library. If multiple drives are installed in the tape library, the top drive is assigned as drive 1. To find out the drive mapping, run the inventory command from TLIB_tool. See **"TLIB_tool" on page 351** for more information.

<slot_num> is the slot number where the tape resides in a tape library. EBR loads the tape from this slot number, when the slot number is specified. Even if the tape serial number is also specified, EBR disregards the tape serial number because EBR gives priority to the slot number. The slot number must be specified if the tape does **not** have a bar code label or a bar code reader is **not** present. Slot number zero (0) is usually reserved for the cleaning tape.

Note EBR_label performs its task according to the tape device type or tape library device type. Make sure you specify the correct type when labeling tapes.

EBR_label Examples for Tape in a Tape Drive

The following example uses the EBR_label utility to prelabel a tape in a standalone tape drive. The example assumes an AIX/6000 system and an 8mm rewinding tape device, /dev/rmt0. A tape serial number of R1 has been selected for this tape and the tape is in volume group VG1.

Insert an unlabeled tape in the drive, and enter the following command at the shell prompt:

EBR_label tape=/dev/rmt0 type=8mm ser=R1 vg=VG1

EBR_label displays the following message when it successfully labels the tape:

```
EBR_label: Tape successfully labeled.
```

An example of the EBR_label command to label a tape is shown below for each supported platform:

AIX

For AIX/6000:

EBR_label tape=/dev/rmt0 type=8mm vg=AIX ser=T19960508

HP-UX

For HP-UX:

EBR_label tape=/dev/rmt/0m type=4mm vg=HPUX ser=T19960508

SOL

For Solaris:

EBR_label tape=/dev/rmt/0m type=8mm vg=SUN ser=T19960508

WIN

For Windows Server:

EBR_label tape=tape0 type=8mm vg=NT ser=T19960508

Use the EBR_tdir command to display the label for a tape.

EBR_label Example for Magnetic Disk

Backing up to a disk file is intended for debugging and trial runs, so backup disk files are treated the same as tape. You must assign each backup disk file a serial number and a volume group name, just as for tape. For disk, EBR_label creates the disk file.

For disk, the first parameter is disk=<file>, where <file> is the full path name of your backup disk file. For example, if your disk file resides in the /fnsw/local/tmp directory and has a file name of BDT960409001, specify your “disk=” keyword as follows:

For UNIX:

disk=/fnsw/local/tmp/BDT960409001

For Windows Server:

disk=\fnsw_loc\tmp\BDT960409001

For disk, the “type=” parameter is not allowed. The serial number and volume group name parameters are the same as for tape.

UNIX

A UNIX command line example for labeling a disk file is:

```
EBR_label disk=/fnsw/local/tmp/BDT960409001 \
   ser=BDT960409001 vg=MONPM3_ONLIVAL
```

WIN

A Windows Server command line example for labeling a disk file in the D drive is:

```
EBR_label disk=D:\fnsw_loc\tmp\BDT960409001
   ser=BDT960409 vg=MONPM3_ONLIVAL
```

EBR_label Example for a Tape Library

The two examples use EBR_label to prelabel a tape in an Exabyte tape library.

If you have a tape without a bar code label or if a bar code reader is not present, you must specify the slot number. The slot number is another method for EBR to identify the tape when there is no bar code label.

If you have a tape with a bar code label, you can identify the tape by one of the following methods:

- Specify the serial number (BAR_xxxxxxxx)
- Specify the slot number.

The first example below of the EBR_label command specifies any Exabyte tape library type.

```
EBR_label lib=/dev/chgr0 lib_type= Exabyte lib_drive=1 \
    tape=/dev/rmt0 type=4mm ser=BAR_00000160 \
    vg=MONDAY
```

The first example shows that the tape library full path name is /dev/chgr0. A 4mm tape will be loaded to tape drive 1. The rewinding tape device name is /dev/rmt0. The volume group name is Monday. The important point in this example is that the tape is bar code labeled. For that reason, the user chooses to specify the tape serial number. The tape serial number begins with BAR and ends with 00000160, which is the same number as the bar code label itself.

The second example below of the EBR_label command specifies an Exabyte 218 (4mm) tape library type.

**EBR_label lib=/dev/chgr0 lib_type="EXB-218" lib_drive=2 \
slot=8 tape=/dev/rmt1 type=4mm ser=T1 vg=MONDAY**

The second example shows that the tape library full path name is also /dev/chgr0. A 4mm tape will be loaded to drive 2. The rewinding tape device name is /dev/rmt1. The volume group name is Monday. This tape is **not** bar code labeled or a bar code reader is **not** present. You must specify the slot number to specify where the tape resides so EBR can load the tape from this slot. In this example, the tape resides in slot number 8. The tape serial number is T1. The tape serial number should not be specified in the BAR_xxxxxxxx format unless the tape is bar code labeled.

Relabeling a Disk File or a Tape in a Tape Drive or Tape Library

EBR_label with the –erasedata option allows you to relabel a disk file or a tape in a standalone tape drive or tape library.

If a backup fails or is canceled, you may retry the backup using the same backup tape or disk file before the backup tape or disk file expires. You must first prepare the tape or disk file to be reused. Use EBR_label with the –erasedata option to modify the expiration date.

The command line syntax if using a standalone tape backup device is:

```
EBR_label tape=<full_path_name> type=<tape_device_type> \
    –erasedata
```

The command line syntax if using a tape library backup device is:

```
EBR_label lib=<full_path_name> lib_type=<lib_device_type> \
    lib_drive=<drive_num> slot=<slot_num>  \
    tape=<full_path_name> type=<tape_device_type> –erasedata
```

The command line syntax if using a disk backup device is:

EBR_label disk=<full_path_name> –erasedata

The EBR_label program, when using the –erasedata option, erases data from the tape or disk file, but preserves the tape serial number, volume group name, and importantly the count of the number of backups.

To relabel a previously used backup tape, we recommend using the –erasedata option instead of specifying a serial number and volume group name. Specifying the serial number and volume group resets the backup count to 0 (zero). Furthermore, a chance of incorrectly typing the serial number or volume group name always exists. we recommend that you specify the serial number and volume group name only once in the life of a tape.

Example of Relabeling Tape in a Standalone Tape Drive

The following is an example command of how to relabel tape in a standalone tape drive using the –erasedata option. The first parameter is tape=<full_path_name>, where full_path_name is the full path name of your tape drive. In this UNIX example, your tape drive resides in the /dev directory and has a file name of rmt0. The tape drive type is 8mm.

EBR_label tape=/dev/rmt0 type=8mm –erasedata

Example of Relabeling Tape in a Tape Library

The following is an example command of how to relabel tape in a tape library using the –erasedata option.

**EBR_label lib=/dev/chgr0 lib_type=Exabyte lib_drive=1 **
slot=8 tape=/dev/rmt0 type=8mm –erasedata

You must specify the slot number in the command syntax. The slot number is necessary for the program to locate where the tape resides. In this example, your tape library has a full path name of /dev/chgr0. The library type is any Exabyte tape library. An 8mm tape is to be loaded from slot 8 to tape drive 1. The tape device name is /dev/rmt0.

Example of Relabeling a Disk File

The following is an example command of how to relabel a disk file using the –erasedata option. The first parameter is disk=<full_path_name>, where full_path_name is the full path name of your backup disk file. In this UNIX example, your disk file resides in the /fnsw/local/tmp directory and has a file name of BDT960409001.

EBR_label disk=/fnsw/local/tmp/BDT960409001 –erasedata

CAUTION When using the –erasedata option or when specifying the serial number and volume group to relabel a disk file or a tape in a standalone tape drive or tape library, EBR_label does not prompt you for confirmation of the correct tape. Specifying the wrong tape can cause permanent loss of the data on the tape.

Troubleshooting EBR_label Problems

EBR_label Failures

If EBR_label fails, one of the following may be the cause:

- The tape cartridge is not enabled for writing
- The tape drive name, tape drive type, or disk file name is incorrect

- Command line parameters contain embedded blank characters

  No embedded blanks are allowed in command line parameters. No spaces can occur immediately before or after an equal sign (=).

Correct the errors and retry the command.

Tape Read Errors

When you specify the –erasedata option, EBR_label reads the tape label, modifies the expiration date, and writes the modified date to the tape. If your tape was originally labeled on a tape drive with a different block size than the block size specified for the current tape drive, tape read errors may occur when you attempt to label the tape. If you want to change the block size of the tape drive, see your operating system documentation for specific instructions for changing the characteristics of your tape drive. However, if you do not want to change the block size, run EBR_label, specifying the original volume group name and serial number of the tape. Do not specify the –erasedata option. This approach relabels the tape with the original volume group name and serial number, resets the backup count to zero, but does not change the original block size.

EBR_orreset

If an Oracle database in online backup mode terminates abnormally, the database is left in a state in which the rollforward of redo logs has not completed. To reset the database to a working state, run the EBR_orreset utility.

The syntax of the utility is:

```
EBR_orreset [pfile = <file_path>]
```

where <file_path> is the full path name of the Oracle parameter file. The default is /fnsw/local/oracle/init.ora (for UNIX platforms) or <drive>:\fnsw_loc\oracle\init.ora (for Windows Server platforms).

CAUTION Do not attempt to run EBR_orreset on a host in which a backup or restore operation is in progress. Doing so causes the backup or restore operation to fail. Contact your service representative before attempting to run EBR_orreset.

If Oracle is online, EBR_orreset issues Oracle Server Manager commands to mark the end of the online backup. If Oracle is offline, EBR_orreset starts Oracle, automatically rolls forward the databases, then shuts down the databases.

EBR_tdir

The EBR_tdir utility displays the tape label on a tape in a standalone tape drive or tape library, or on a disk file.

After the label is displayed, EBR_tdir ejects the tape as the default. To keep the tape in a standalone tape drive, specify the “eject=no” parameter. A tape in a tape library is always both ejected and unloaded.

If run without parameters or with incorrect parameters, EBR_tdir displays an extensive help message.

EBR_tdir performs its task according to the type of tape you are using. Make sure you specify the correct type.

Tip Insert the tape into the tape drive before you start the EBR_tdir utility. If the tape is not in the drive when EBR_tdir starts, the program terminates with an error message indicating the tape drive is offline.

You can use EBR_tdir to determine when to retire the tape. The “number_of_backups=” field in the EBR_tdir output displays the number of backups that have been written to this tape. However, the backup count is reliable only if you initially labeled the tape with EBR_label and subsequent relabelling has been performed with the EBR_label –erasedata option. Running EBR_label without the –erasedata option resets the backup count to 1, making it impossible for you to determine the correct number of backups.

Command Syntax

The command line syntax for displaying the label of a tape in a standalone tape drive is:

```
EBR_tdir tape=<full_path_name> type=<tape_device_type> \
    [eject={yeslno}]
```

The command line syntax for displaying the label of a tape in a tape library is:

```
EBR_tdir lib=<full_path_name> lib_type=<lib_device_type> \
    lib_drive=<drive_num> \
    {barcode=<barcode_name> I slot=<slot_num>}
    tape=<full_path_name> type=<tape_device_type>
```

The command line syntax for displaying the label of a disk file is:

```
EBR_tdir disk=<full_path_name>
```

where

<full_path_name> is the full path name of the rewinding tape drive, either in a standalone tape drive or in a tape library (for example, /dev/

rmt0.4), or of the disk file. For tape, use the highest tape density setting available.

Note EBR checks both the rewind and non-rewind tape device names and selects the appropriate device. However, the EBR_tdir and EBR_label utilities require that you specify only one tape device name. You can specify either the rewind or no-rewind name.

<tape_device_type> See **"Initial Tape Labeling" on page 334** for a description of the tape device types.

eject={yes|no} is an option that instructs EBR_tdir to eject the tape or leave the tape inside the standalone tape drive after displaying the tape directory. The default is yes (to eject the tape). This option does not apply to tape in a tape library.

<lib_device_type> specifies the tape library type and is one of the following keywords. The keyword may be enclosed in double quotes, except for the Exabyte keyword, which does not require double quotes.

Keyword Value	Description
"EXB-210"	Exabyte 8 millimeter cartridge tape library
"EXB-218"	Exabyte 4 millimeter cartridge tape library
"EXB-220"	Exabyte 8 millimeter "Mammoth" cartridge tape library
Exabyte	Refers to any of the above Exabyte tape libraries and can be used if you are not sure of the type

<drive_num> is tape drive location in the tape library. If multiple drives are installed in the tape library, the top drive is assigned as drive 1. To find out the drive mapping, run the inventory command from TLIB_tool. See **"TLIB_tool" on page 351** for more information.

<barcode_name> is the bar code label of the tape to be read. The bar code name may be specified if the tape has a bar code label and a bar code reader is present. Note that the bar code is not the volume serial number.

<slot_num> is the slot number where the tape resides. EBR loads the tape from this slot number, when the slot number is specified. The slot number must be specified if a tape does **not** have a bar code label or a bar code reader is **not** present. If both the slot number and bar code name are specified, EBR disregards the bar code because EBR gives priority to the slot number.

Tape Label Display Examples

For Tapes in Standalone Tape Drive

The following EBR_tdir examples display the labels of the 8mm or 4mm tapes labeled in the EBR_label command examples **page 338**:

AIX

For AIX/6000:

EBR_tdir tape=/dev/rmt0 type=8mm

HPUX

For HP-UX:

EBR_tdir tape=/dev/rmt/0m type=4mm

SOL

For Solaris:

EBR_tdir tape=/dev/rmt/0mb type=8mm

WIN

For Windows Server:

```
EBR_tdir tape=tape0 type=8mm
```

Note You must specify the same tape device type in both the EBR_label command and your backup script. Failing to do so may result in an inability to list the backup tape directory with EBR_tdir or an inability to restore from the backup tape. See **"Selecting Tape Device Type" on page 132** for more information.

For Tapes in Tape Library

The following EBR_tdir examples display the labels of the 4mm tapes in an Exabyte 218 tape library:

The first example specifies a slot number of 8 because the tape has no bar code and a bar code reader is not present. The tape library full path name is /dev/chgr0. A 4mm tape will be loaded to tape drive 1 from slot number 8. The tape device name is /dev/rmt0.

```
EBR_tdir lib=/dev/chgr0 lib_type=Exabyte lib_drive=1 \
    slot=8 tape=/dev/rmt0 type=4mm
```

The second example specifies the tape library type as an Exabyte 218 and the bar code label as 000002. Because a bar code label is specified, the slot number is not required. The tape library full path name is /dev/chgr0. A 4mm tape will be loaded to tape drive 2 from the slot specified by the bar code. The tape device name is /dev/rmt0.

```
EBR_tdir lib=/dev/chgr1 lib_type="EXB-218" lib_drive=2 \
    barcode=000002 tape=/dev/rmt0 type=4mm
```

Sample Output

The following sample output displays the label for the diskdata.bu1 disk file:

```
corona> EBR_tdir disk=diskdata.bu1

Volume label:
EBR_volume_label
volume_label_format=1
volume_serial=R1
volume_group=VG1
number_of_backups=8
volume_state=inuse
number_of_data_blocks=32
number_of_ECC_blocks=8
backup_start=835292505 Thu Jun 20 10:41:45 1998
expiration=835292507 Thu Jun 20 10:41:47 1998
number_of_threads=1
.................................................
number_of_volumes=1
volume_serial=R1
number_of_datasets=4
dataset_name=sec MKF (1 of 1)
location=TapeServer1:bandit:FileNet
offline_full_backup
dataset_name=part_perm partition (1 of 1)
location=bandit
dataset_name=tran MKF (1 of 1)
location=TapeServer1:bandit:FileNet
offline_full_backup
dataset_name=pc1_f cache (1 of 1)
location=page_cache1:bandit:FileNet
offline_full_backup
end_volume_label
```

EBR_ulmk

EBR_ulmk unlocks MKF databases that have been locked for an offline backup that did not complete. Locking the database insures a clean backup by keeping the database offline for the duration of the backup.

EBR_ulmk cannot unlock an MKF database that is being restored. The restore must complete successfully.

The syntax of the EBR_ulmk utility is:

EBR_ulmk <MKF_base_data_file_name>

where <base data file name> is the full path name of the base data file for the MKF database you wish to unlock.

The following command example unlocks the permanent database:

EBR_ulmk /fnsw/dev/1/permanent_db0

TLIB_tool

The TLIB_tool is used to control Exabyte tape library functions. The TLIB_tool allows you to initialize or reset the tape library, lock or unlock the front access door, acquire inventory information about the tape library, move or load or unload media, position the tape library arm, eject media, and search or move media by bar code label.

TLIB_tool is supported for the Exabyte 210, Exabyte 218, and Exabyte 220 tape libraries on the AIX/600 platform and for the Exabyte 218 tape library on the HP-UX platform. TLIB_tool only operates in command line mode. Do not use TLIB_tool in a script for batch processing.

Note EBR does not support the Exabyte tape library on Solaris and Windows Server platforms.

Start the utility by entering the command from the system prompt:

TLIB_tool

Respond to the following prompts:

Enter tape library name:
Enter tape library type:

For tape library name, enter the device name. The device name must include the path name, for example, /dev/chgr0.

For tape library type, enter one of the following:

- EXB-210
- EXB-218
- EXB-220
- Exabyte (the tape library automatically finds the correct type from firmware if you do not know the device type.)

A list of commands used to control tape library functions will display. Enter the appropriate command at the system prompt, for example:

> inventory

Respond to any prompts that follow.

Enter **quit** to exit the TLIB_tool program.

If you need a more detailed description of each tape library command, read the following information.

init Command

Use init to initialize the tape library, including the tape library internal elements, storage slots, and the transport device (arm). The init option scans the tape label and checks for the presence of media in the tape library. It checks the accessibility of each internal element and for the presence of a bar code reader. Running an initialization can generate updated information for an inventory command. The initialization may take several minutes to complete depending on the size and configuration of the tape library.

reset Command

Use reset to reset the tape library. Reset causes a hardware reset of the device, equivalent to a power cycle. When you must do a power cycle, the reset command is preferable to a power cycle. The reset command will unlock the tape library front access door and can serve as an emergency lock override in case the unlock command fails.

lock Command

Use lock to activate a mechanical locking mechanism, if present, on the front access door of the tape device. When the front access door is locked, you cannot insert media into or remove media from the tape device, even with a key.

unlock Command

Use unlock to activate a mechanical unlocking mechanism, if present, on the front access door of the tape device. When the front access door is unlocked, you can insert media into or remove media from the tape device.

inventory Command

Use inventory to return information about the tape library, including library type, hardware configuration, slot map, and drive map. You usually need the inventory information as input for other TLIB_tool commands.

An example of information returned on an Exabyte 218 tape library by the inventory command is illustrated below.

```
Product Type          :EXABYTE EXB-218 Revision 4.11
Driver               :CHGR Version 1.0 by Exabyte Corp.
Number of Drives      :2
Number of Slots       :19
Number of Transports :1
Barcode Reader        :Present
Open Count            :1
Unit locked              :No
```

Type	Id	Addr	Status	Accessible	Label
Slot	0	0	Full	yes	
Slot	1	1	Full	yes	
Slot	2	2	Empty	yes	
Slot	3	3	Empty	yes	
Slot	4	4	Empty	yes	
Slot	5	5	Empty	yes	
Slot	6	6	Full	yes	000004
Slot	7	7	Empty	yes	
Slot	8	8	Empty	yes	
Slot	9	9	Empty	yes	
Slot	10	10	Full	yes	000003
Slot	11	11	Empty	yes	
Slot	12	12	Empty	yes	
Slot	13	13	Full	yes	000002
Slot	14	14	Full	yes	000001
Drive	1	82	Empty	yes	
Drive	2	83	Empty	yes	

The following table describes the information displayed in the illustration above by the Inventory command:

Information	Description
Product Type	Tape library model and revision numbers.
Driver	Tape library driver name and driver version number
The FileNet required **Exabyte** tape library driver	Versions are:AIX/6000Version 1.0 HP-UXVersion 1.3 An error message displays if the driver version loaded is earlier than the supported FileNet version. Do not use an earlier driver version. A warning message displays if the driver version loaded is later than the FileNet required version. See **"Tape Library Support" on page 31** for more details on drivers.
Number of Drives	The number of tape drives.
Number of Slots	The number of storage slots, including slot 0. Slot 0 is usually reserved for the cleaning tape.
Number of Transports	The number of robotic transport arms.
Barcode Reader	Indicates whether a bar code reader is present or not present.
Open Count	Indicates number of processes currently accessing the tape library.
Unit locked	**Yes** indicates the front access door is locked. **No** indicates the front access door is not locked.
Type	Slot indicates a storage slot. Drive indicates a tape drive.
Id	Indicates the slot number or drive number ID.
Addr	Driver's element address. This is used for internal reference.
Status	**Full** indicates that the tape resides in the slot or tape drive. **Empty** indicates that no tape resides in the slot or tape drive.
Accessible	Indicates that the slot or drive is available for a load or unload or move operation.
Label	Indicates a bar code label is present on the tape when a number, such as 000004, is displayed. A blank indicates no bar code label is present.

move Command

Use move to move media from one storage slot to another storage slot. To move media from a storage slot to a tape drive, use the load command. To move media from a tape drive to a storage slot, use the unload command.

After entering the move command, reply to the following prompts:

Move media from slot:
To slot:

position Command

Use position to move the tape library's transport device (arm) in front of the specified storage slot or tape drive.

After entering the position command, reply to the following prompt:

Position transport to slot [s] or tape drive [d]:

Enter either **s** or **d**, depending on whether you are positioning the arm in front of a storage slot or tape drive.

Respond to either prompt:

Slot id :
Drive id :

load Command

Use load to move media from a storage slot to a tape drive.

After entering the load command, reply to the following prompts:

```
Load media from slot :
To tape drive :
```

unload Command

Use unload to move media from a tape drive back to the storage slot from where the media was loaded. The program knows to which storage slot to move the media. The unload command can only be used when the tape has been automatically ejected by an application program or when the tape is an Exabyte cleaning cartridge, which self-ejects.

After entering the unload command, reply to the following prompt:

```
Unload media from tape drive :
```

eject Command

Use eject to discharge and unload media from the tape drive. Unloading the media moves the media from a tape drive back to the storage slot from where the media was loaded.

After entering the eject command, reply to the following prompts:

```
Eject and unload media from drive (specify drive id) :
Specify tape device file name :
```

Make sure you enter the full path name for the tape device file name.

search Command

Use search to search for media containing a bar code label. This command returns the element type (slot or drive), status (full or empty), and accessibility (yes or no) information. If the tape library does not support a bar code reader, the search command fails.

After entering the search command, reply to the following prompt:

```
Enter barcode :
```

An example output for barcode, 000005, displays as follows:

Type	Id	Addr	Status	Accessible	Label
Slot	**10**	**10**	**Full**	**yes**	**000005**

loadbarcode Command

Use loadbarcode to move media from a storage slot to a tape drive based on the media's bar code label.

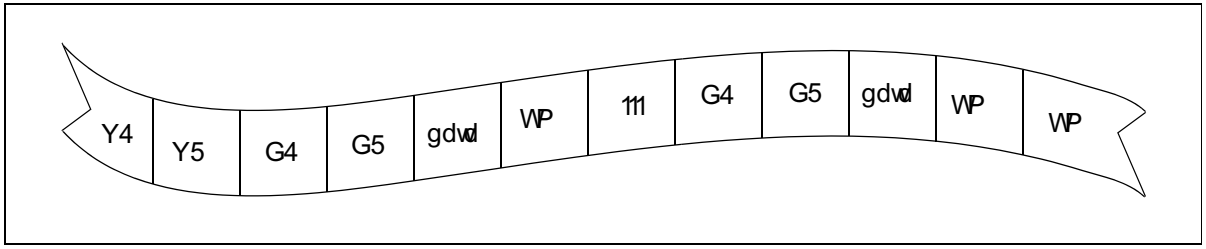
After entering the loadbarcode command, reply to the following prompts:

```
Enter barcode :
To tape drive :
```

Appendix B – Tape Format

All tape blocks are 32,768 bytes long. Every tape block starts with a common header that includes a 256-bit checksum, a 64-bit block sequence number, a block type, etc.

The global format of information on the tape is shown in the following illustration:



The volume label, <V1>, is written to the tape first. A second copy of the volume label, <V2>, follows.

The volume labels are followed by a duplicated dataset label, <D1> and <D2>, for every dataset specified in the script.

The dataset labels are followed by the compressed data blocks, <data>, of the dataset. Each dataset ends with a tape mark, <TM>.

The ellipsis (...) in the illustration represents zero or more occurrences of:

<D1><D2><data><TM>

An extra tape mark is written at the end of the tape so two tape marks, <TM><TM>, always occur at the end of a tape.

The volume labels and dataset labels are generally ASCII characters, so you can view them with MKF_dump and other utilities, if necessary. For help in using MKF_dump, contact your service representative.

Unlike actual tape marks on tape, tape marks on disk are 32K blocks of type “tape mark.”

EBR uses the high speed search capability of the tape drive to advance to the next tape mark when searching for a dataset to restore. Because of the difference between tape marks on tape and simulated tape marks on disk, EBR requires that if you back up to tape, you must restore from that tape. Similarly, if you back up to a disk file, you must restore from that disk file.

You can interchange tapes between systems of the same type (for example, between two AIX/6000 systems). Tapes generated by EBR cannot be interchanged between different platforms. For example, 4mm DAT tapes generated on an HP-UX system contain inter-record gaps that cannot be read by an AIX/6000 system or a Solaris system.

If you plan to interchange EBR tapes between like systems, the tape drive block size on each of those systems must be set to the same value. For AIX/6000 systems, we recommend that you set block size to zero (0=variable block size) for all tape drives.

In addition, the tape drives you use to create the tape and read the tape must be configured with the same characteristics (for example, density). Then any EBR tape can be used on any of the same-type systems.

To determine the current block size setting for AIX/6000, use the following command:

mt –f <tape_device_name> status

For other operating system platforms, see your operating system manuals or contact your operating system administrator for equivalent commands.

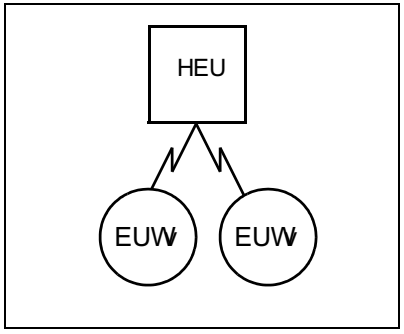
Appendix C – EBR Program Operation

Information in this appendix is not required to operate EBR. However, the information provides some background about how EBR works internally to process information and initiate work. In addition, the information may help you to interpret data written to the progress log.

EBR Processes

The EBR program is the **master process**. EBR orchestrates concurrent backup and restore, that is, to different systems using multiple tape drives. EBR performs a remote procedure call (RPC) to each tape thread (one thread exists per tape cartridge). An RPC requires a surrogate process, referred to as a **request handler process**, to run on the target system.The request handler process can run on the same host as that on which EBR is running or on a remote host in the network.

In the following diagram, the EBR program (the master process) issues an RPC to each of the two available threads (in this example, two tape cartridges are available and, since each cartridge must have its own thread, two threads are shown).



The BRTs program is the **tape request handler process** for the transparently remotable abstract (shared library). To minimize the amount of data that is transmitted over a network connection, mainly status information is transmitted over the connection between EBR and BRTs.

Since BRTs is usually busy responding to requests for status (status RPCs) issued by the EBR program, BRTs forks two processes (called **offspring processes**) to actually do the backup to or restore from tape. One offspring process writes data to tape (for backup) or reads data from tape (for restore). The other offspring process issues an RPC to the host containing the disk dataset that is being backed up or restored and computes ECC information. The RPC initiates the read operations from the disk dataset (for backup) or write operations to the disk dataset (for restore).

The three processes described above (the tape request handler process and the two offspring processes) exist on the host with the tape drive. One of the following letters is assigned to each process so that you can distinguish in the progress log the status messages for each process:

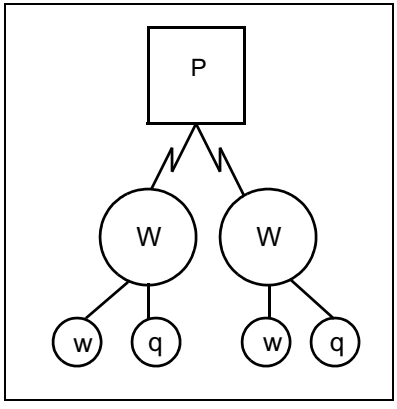
Process Designator	Description
T	Assigned to the tape request handler process. The letter is capitalized because this process is the parent of the two offspring processes, t and n.

t	Assigned to the process that actually reads or writes tape (the “tape worker process”).
n	Assigned to the process that generates RPCs to the host where the disk dataset resides. Referred to as the network disk client process, this process also verifies the checksum of tape blocks and computes the error checking and correction (ECC) blocks. The CPU activity of the network disk client process is overlapped with the tape I/O activity of the tape worker process.

The tape request handler process, the tape worker process, and the network disk client process communicate using shared memory and FileNet interlocks.

When you examine a progress log, you see the letter M assigned to the master (EBR) process. In the following diagrams, EBR is identified with the letter M also.

Expanding the diagram and using the letters to stand for the processes results in the following illustration:



The network disk client process calls the transparently remotable data abstract BRD to generate an RPC to the host containing the disk dataset to be backed up or restored. The BRDs program is the request handler for BRD.

Since the disk request handler process is busy responding to RPCs for disk blocks and status, it forks two offspring processes to actually perform the disk I/O and software compression.

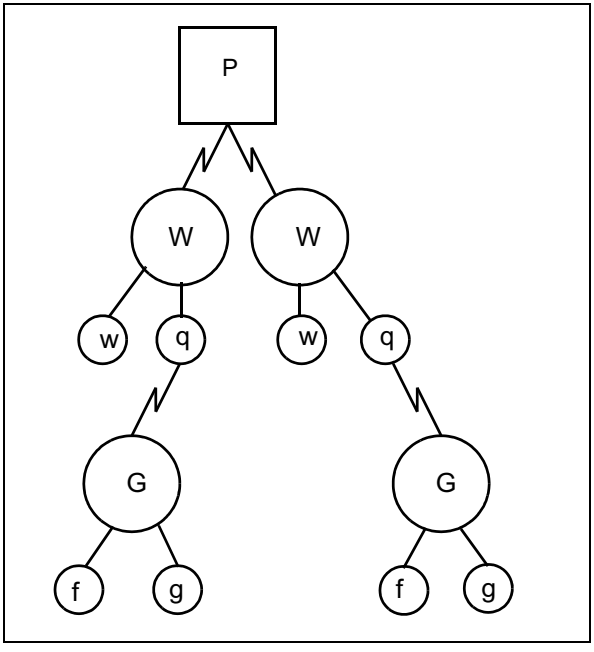
The letters D, d, and c are assigned to the disk request handler's offspring processes.

Each disk request handler offspring process is described in the table below.

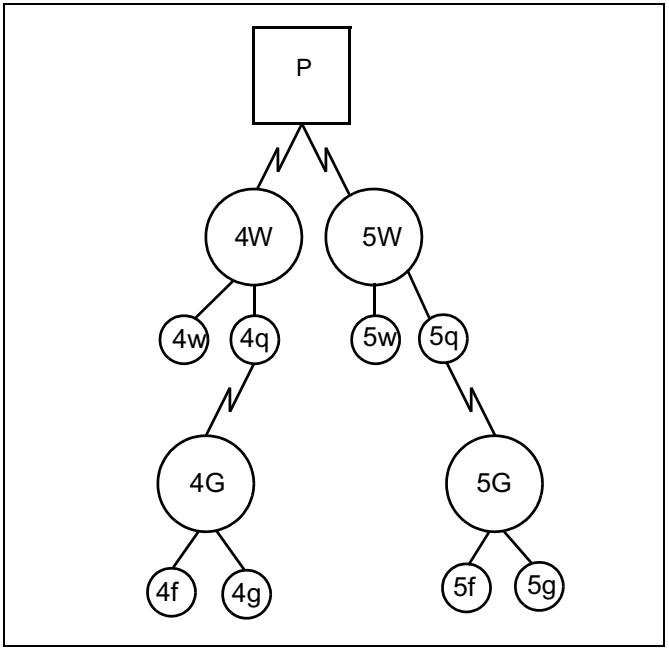
Process Designator	Description
D	Assigned to the disk request handler process. The letter is capitalized because this process is the parent of the two offspring processes, d and c. The disk request handler process responds to RPCs from the network disk client process. Requests for disk blocks and status alternate: the network disk client process gets (for backup) or sends (for restore) a block of compressed data; then asks for a block of status; then gets or sends a block of compressed data; and so on.
d	Assigned to the disk worker process, the process that actually reads to or writes from disk. For backup, the disk worker process sends buffers of uncompressed disk data to the compressor/decompressor process. For restores, the disk worker process gets buffers of uncompressed disk data from the compressor/decompressor process.
c	The compressor/decompressor process exists so that disk I/O can be overlapped with compression/decompression.

The disk request handler process, the disk worker process, and the compressor/decompressor processes communicate using shared memory and FileNet interlocks.

The expanded diagram below shows these additional processes:

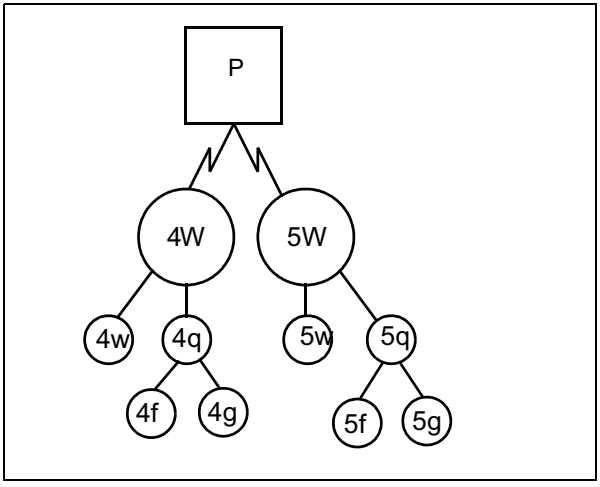


Inserting the thread numbers before the process letters results in the following diagram:



The previous two diagrams do not show network boundaries.

If the tape device and the disk dataset are on the same host, EBR optimizes its operation by eliminating the disk request handler process and making normal (local) procedure calls instead of remote procedure calls. This action is shown in the following diagram:



High Performance Operations

Since four worker processes (the tape worker, the network disk client, the compressor/decompressor, and the disk worker) exist for each tape drive, the backup or restore is highly pipelined, resulting in high throughput.

Only compressed data is sent over the network, increasing performance. EBR verifies the checksum value and, if the block sent was corrupted, resends the block. ECC blocks are not sent over the network; rather, they are computed (for backup) or checked (for restore) by the network disk client process.

The compressed blocks sent over the network are exactly the same format as the blocks written to tape. Each compressed block is 32K (32,768) bytes in size. See **"Appendix B – Tape Format" on page 360** for details of the tape block format.

Some disk data compresses extremely well. For example, 128K-byte chunks of all-zero blocks compress 2,730 to 1.

Status Message Overhead

Producing status messages involves some overhead for the system. In general, the master process polls for status every second, while the disk worker and the tape worker (the two processes that generate most of the status messages) produce status about every four seconds. This approach strikes a balance between providing responsive status and reducing the overhead required to produce the status messages for large amounts of data being sent over the network.

Appendix D – Byte Ordering Issues

Byte ordering is important for you to understand if you use Enterprise Backup/Restore to back up or restore files across different types of hosts, as you might have in a large enterprise. This appendix describes:

- Byte ordering
- The two types of byte ordering you may encounter
- How EBR operates with these byte order types

Byte Order Definition

Byte order refers to the method used by a CPU to transfer a word of data to an I/O device. A longword (also called simply a “long”) is four bytes; a shortword (a “short”) is two bytes.

The CPU determines the byte ordering. When transmitting the long-word to an I/O device, the CPU shifts bytes of data out from one end or the other of the longword. Big-endian byte ordering indicates that the bytes of the longword are shifted out of the most significant end of the longword; little-endian indicates that the longword bytes are shifted out of the least significant end. (See **“Byte Order Independence” on page 61** for more information.)

The following CPUs are big-endian: MC 68040 (and 68030, etc.), IBM RS 6000, Power PC, Sun SPARC. The following CPUs are little-endian: DEC VAX, Intel 80486 (and 80386, etc.), Pentium.

In general, when sending a file between hosts of different byte order, the CPU swaps the bytes of the longs and the shorts into reverse order. However, transfers of arrays of bytes (for example, an ASCII string) maintain the same order.

Multiple Byte Order Systems

EBR accommodates both byte orderings. The simplest case for EBR to deal with is when the system is all big-endian or all little-endian. A more complex case exists when the systems in the enterprise are of mixed byte order. In a mixed byte order system, the host to which the tape drive is connected may either match or not match the byte order of the host on which the disk dataset resides. Furthermore, the byte order of the host to which the tape drive is connected during a restore may either match or not match the byte order of the host on which the backup tape was written. Although this mixed byte order system is more complex, EBR readily addresses the data transfer requirements of such a system.

The byte order of the system with the tape drive does not have to match the byte order of the system being backed up. Furthermore, the system that does the restore does not need to have the same byte order as the system on which the backup was made.

Coupled with the ability of threads to share tape drives, EBR's byte ordering independence means that you can always perform a backup or restore as long as (1) at least one tape drive of the appropriate type is working on some host in your enterprise and (2) that host can be reached by a FileNet RPC.

Performance Factors

In a mixed byte order environment, the cost of performance is shared by the hosts owning the tape drives and the hosts owning the disk datasets. The CPU performance cost of mixing byte orders is not significant.

Conversion Issues

You can, for some dataset types, restore the data to a system with a different byte order than that on which it was backed up. However, except for MKF databases, the dataset may be unusable until additional conversion is performed. The additional conversion procedures are dependent on dataset type and are outside the scope of EBR.

Using EBR, you can restore an MKF database to a system with a different byte order, different file names, different number of files, etc., because MKF is designed with this flexibility in mind. MKF databases are easily and quickly converted during restore.

In contrast, the design of other RDBMS databases (for example, Oracle) is such that they cannot be restored on systems with opposite byte order or different file layouts. As a result, EBR cannot restore the FileNet index database (which is managed by your RDBMS) to a system of a different byte order or different file layout.

If you restore a partition or file (with the exception of a file that is a character stream) to a system with a different byte order and that file contains any shortwords or longwords, you will have to run a conversion application to convert the shorts and longs to the new byte order. Such a conversion task is outside the scope of EBR.

Appendix E – Computing MKF Log Space

In preparing your system for regular backups, you need to determine how much magnetic disk data space to allocate for MKF recovery log files. This appendix describes the procedure you can use to determine this value.

In the following procedure, we calculate magnetic media space requirements for the permanent database first. Use the same steps to determine the space allocation requirements for any of the other MKF databases, substituting the correct database name.

1 Run the MKF_dump utility against the permanent database.

UNIX **MKF_dump /fnsw/dev/1/permanent_db0** for UNIX platforms

WIN **MKF_dump \fnsw\dev\1\permanent_db0** for Windows Server platforms

2 Enter **1** to display the control block of the permanent database.

Two fields in the display of the control block, **most sig aij bsn** and **least sig aij bsn**, form a 48-bit after image journal block sequence number (bsn). The most significant part is 32 bits in length; the least significant part is 16 bits in length. (The integer is stored in two parts because the word width of the CPU is only 32 bits.) Logically, the two parts form a single 48-bit twos complement integer. The value of each part of the bsn is displayed in both hexadecimal and in decimal.

3 Record the current time and the after image journal bsn.

4 Enter **q** to quit the control block display.

5 Enter **q** to quit the MKF_dump program and **y** at the confirmation prompt.

6 Exactly 24 hours later, perform Steps 1 and 2 again.

Record the current after image journal bsn.

7 Calculate the amount of data written to the permanent database recovery log.

a Subtract the after image journal bsn value you recorded in Step 3 from the after image journal bsn value you recorded in Step 6 to get the number of 1024-byte blocks written to the MKF recovery log in 24 hours.

b Divide that value by 1,000 to obtain the number of megabytes of log data produced in 24 hours.

c Multiply that number by at least 2 or 3. The result is the minimum amount (in megabytes) of disk space you should allocate to the MKF recovery log for the permanent database.

8 Perform Steps 1 through 7 to determine the amount of recovery log disk space you need for the security database.

Use the full path name of the security database in your MKF_dump command, for example:

UNIX /fnsw/dev/1/sec_db0 for UNIX platforms

WIN <drive>:\fnsw\dev\1\sec_db0 for Windows Server platforms

Note Do not be concerned about the size of the recovery log for the transient or NCH databases. The transient database must be perfectly synchronized with the page cache. Since the page cache has no rollforward capability, rolling the transient database forward serves no purpose. Therefore, you do not need to back up or restore a transient database recovery log file.

The NCH database is small and can easily be recreated by means other than rollforward. (See **"Recreating the NCH Database" on page 95**.) You should never have to back up or restore the NCH database.

If you have questions about this procedure or need help in calculating MKF recovery log disk space, contact your service representative.

Appendix F – Enabling Archive Log Mode

Some backup and restore strategies require that you enable archive log mode for your RDBMS transaction logs. This appendix provides an overview of these logs and a procedure for enabling archive log mode.

Note EBR can only back up Oracle databases used by FileNet Image Services and eProcess.

Archive Logs Overview

Archive logs are copies of the RDBMS transaction logs (for example, the Oracle redo logs). During system installation, you can set a FileNet configuration parameter to enable archive logging, or you can set this parameter at any time using the FileNet System Configuration Editor. The procedure to enable archive logging consists of:

- Configuring the FileNet software for archive logging
- Enabling the RDBMS to write archive logs to a selected directory

When you have enabled the FileNet **and** RDBMS archive logging parameters, your RDBMS can write to these archive logs.

If you choose to archive logs in your system, your service representative can enable archive logging on your system when your FileNet software is installed and configured. You can also use the instructions in this appendix to enable archive logging at a later time.

CAUTION Do not randomly turn archive log mode off and on. Doing so may prevent a full recovery of RDBMS databases during a restore. Select the archive log mode necessary for your environment and leave the mode at that setting.

Enabling Oracle Archive Logging

Archiving Oracle redo logs is optional if:

- you perform offline backups and
- recovering the database by restoring to the last backup is an acceptable level of recovery for your system.

However, archiving the Oracle redo logs is required if you choose to perform online backups.

CAUTION Monitor the Oracle index database regularly. The Oracle RDBMS hangs if insufficient space is available to create the archive log.

Be absolutely sure you have enough magnetic disk space to hold the archive logs until you copy them to tape. You can create a file system for this purpose alone.

The amount of space you allocate to the archive logs is dependent on daily database activity, such as the number of committals, deletions, and updates.

A qualified service representative or a System Administrator can perform the following procedure to enable automatic archiving.

1. Log in as fnsw or as a user that is a member of the fnadmin and dba groups.

2 Create a directory for the Oracle archive logs.

Use the following sequence of commands to create the directory and set the appropriate permissions. (In these sample commands, the /fnsw/local/archivelogs directory is used on a UNIX platform.)

mkdir /fnsw/local/archivelogs

chmod 777 /fnsw/local/archivelogs

chown fnsw /fnsw/local/archivelogs

chgrp dba /fnsw/local/archivelogs

3 Start the FileNet System Configuration Editor.

UNIX On UNIX platforms, enter at the system prompt:

fn_edit

WIN On the Windows Server platform, do the following:

From the Taskbar, click the Start button, point to Programs, point to the FileNet Image Services Configuration folder, and double click the Configuration Editor icon.

4 When the Open Configuration Database dialog box displays, click OK.

5 Select the Relational Databases tab in the FileNet Image Services System Configuration Editor dialog box.

6 Select the Oracle tab.

To **enable** archive logging, click on the pane under the Log Archive Start column. A checkmark appears.

To **disable** archive logging, click on the pane under the Log Archive Start column. The checkmark disappears and an X appears. Log Archive Start works as a toggle.

7 When enabling archive logging, you must enter the directory name to which you want the archive logs created in the Archive Log Destination column.

In the Archive Log Destination box, enter the full path name for the archive log file. (It is not recommend to archive to a tape device.)

You must supply the file name prefix ARCH to Oracle for the UNIX platform. Oracle on Windows Server does not use a file name prefix.

UNIX Oracle for UNIX example: /fnsw/local/archivelogs/arch

WIN Oracle for Windows Server example: \fnsw_loc\archivelogs

Oracle creates the archive logs using this file name and appends a number sequence, similar to:

UNIX Oracle for UNIX: /fnsw/local/archivelogs/arch1_26.dbf

WIN Oracle for Windows Server: \fnsw_loc\archivelogs\ARC00026.001

8 Select Exit from the File menu and click on **yes** in the pop-up dialog box to save the current configuration database and terminate fn_edit.

9 Execute the fn_util updaterdb program to automatically perform the enabling or disabling archive logging steps.

At the system prompt, enter:

fn_util updaterdb

The fn_util updaterdb program will shut down the FileNet software, apply the change directly to the Oracle database, and then restart the Oracle software.

Note Make sure your Oracle parameter file, init.ora, is set to automatically start archive logging. Restarting Oracle reinitializes the Oracle instance with the settings in init.ora, which may not be set for automatic archiving. To verify the correct setting, change to the directory that contains init.ora and use one of the following techniques to display the file contents:

For UNIX: enter the command **less init.ora**

For Windows Server: enter the command **type init.ora**

In a Windows Server environment, you can also use Notepad or a text editor to display the contents of init.ora.

Check for the log_archive_start=true setting in the parameter file. If the parameter is set to false, rerun Steps 2 through 5. For more information on the Oracle parameter file, see your *Oracle Server Administrator's Guide or Oracle Server Backup and Recovery Guide*.

---

10 As any member of the **dba** group, such as **oracle** or **fnsw** user, check to make sure your changes were accepted by entering the following SQL*Plus commands:

sqlplus "/ as sysdba"

(The double-quotes are required; the space between the **/** and **as** is optional.)

The prompt changes to SQL>. Enter the commands in the following steps from this prompt.

11 Start and mount the database using the appropriate initialization file by entering the following command:

UNIX

- For UNIX platforms:

  startup pfile=/fnsw/local/oracle/init.ora mount

WIN

- For Windows Server platforms (substitute the correct drive on your system for <drive> in the following command):

  startup pfile=<drive>:\fnsw_loc\oracle\init.ora mount

Oracle SQL*Plus returns messages similar to the following:

```
Oracle instance started.
Database mounted.
```

12 Verify Archive destination and Automatic archival settings.

Archive destination must be set to the path you established in the Configuration Editor and automatic archival must be enabled. Enter the following command to verify this information:

archive log list;

Oracle SQL*Plus displays information similar to the following. (The output below is based on the UNIX example used in this procedure; your output should reflect the path you establish. For example, the

Windows Server archive destination does not include the arch prefix required by Oracle in UNIX environments.)

```
Database log mode              Archive Mode
Automatic archival             ENABLED
Archive destination            /fnsw/local/archivelogs/ARCH
Oldest online log sequence     16
Next log sequence to archive   17
Current log sequence           17
```

13 Shutdown Oracle and exit SQL*Plus using the following instructions:

a At the Oracle SQL*Plus prompt, enter:

SQL> **shutdown normal**

The following messages confirm the shutdown:

Database closed.
Database dismounted.
Oracle instance shutdown.

b Terminate SQL*Plus with one of the following commands:

SQL> **exit**
or
SQL> **quit**

14 Start Image Services using the task manager or the **initfnsw -y restart** command. Check for errors in the error log after Image Services has started.

Appendix G – Memory Requirements

This appendix describes the following:

- Minimum memory requirements for a FileNet system
- Guidelines for running EBR in a minimally configured memory environment
- Memory requirements for EBR processes

Minimum Memory Requirements

The EBR subsystem consists of approximately three megabytes of code. The code is loaded once into memory then shared by all processes executing the code.

In general, to run the FileNet server software, your system should have at least 64 megabytes of main memory. If you are running EBR on a system with 32 megabytes of memory, you are limited to running two EBR threads concurrently.

EBR Process Memory Requirements

EBR has one master process, three tape processes per thread, and three disk processes per thread. The following local memory requirements were obtained using the **ps –l** command for UNIX platforms and the Memory Usage field of the Task Manager display for the Windows Server platform, then adding the virtual memory size of the kernel stack and the process data segment. The actual number of pages physically resident in memory may be significantly less than the virtual

memory size. Local memory requirements vary between operating system platforms.

The local memory requirements for the master process is approximately 418 KB. The total local memory requirement for the tape processes is approximately 2 MB. The total local memory requirement for the disk processes is approximately 3 MB.

The total shared memory required for the three tape processes necessary for a single thread is approximately 2.9 MB.

The total shared memory required for the three disk processes necessary for a single thread is approximately 0.7 MB.

Tape processes run on the host that owns the tape drive; disk processes run on the host that owns the magnetic disk. These may or may not be the same host. You achieve the most effective use of memory when the disk and the tape are on the same host, especially if multiple threads run on that host.

Memory Requirements for Other System Components

The requirements described in **"EBR Process Memory Requirements" on page 384** are the incremental memory requirements for EBR. When backing up databases and magnetic disk caches, you also incur the memory requirements for active database management subsystems such as MKF and Oracle. The bulk of the memory requirements for the DBMSs are for buffer pools. Buffer pools are in shared memory.

CAUTION If EBR cannot allocate enough shared memory at startup, EBR processes fail and EBR terminates. You must recycle (stop and restart) FileNet software to continue. Save any core dump files and error mes-

sages and contact your service representative for troubleshooting assistance.

Appendix H – Running SNT_update

Occasionally, you may need to update the scalar numbers table (SNT) after a restore operation of the MKF permanent database. This appendix describes the scalar numbers table and provides an update procedure.

Preventing Duplicate Document Numbers

The scalar numbers table (SNT) of the MKF permanent database keeps a record of the next available numbers for several Image Services and document services number spaces. One of these numbers is the next available document ID. Whenever a system operator enters a new document into the system, the Image Services system assigns it a document ID using an increasing number sequence. It obtains the next available number from SNT when creating a new document or batch of documents.

If a magnetic disk crash occurs that results in the loss of the permanent database, it also loses the SNT. This requires restoring the permanent database from the last available backup tape. (A full restore of the permanent database overwrites the current values in the SNT with old values from the SNT of the restored system.) It uses any available transaction logs to roll the database forward to the most current transaction. However, it may not always be possible to roll forward to the exact moment of the crash so the permanent database may become unsynchronized with the index database and documents on optical media. If this unsynchronized condition occurs, the next available document ID in the SNT may be well below what it should be. After the

operator restarts the FileNet software, the system may duplicate document IDs when scanning in new documents.

To prevent assigning duplicate document IDs, the system refers to the scalar numbers table checkpoint file, snt.chkpt, located in the /fnsw/local/sd directory for UNIX platforms or \fnsw_loc\sd directory for Windows Server platforms. As shown in the example below, this checkpoint file contains a backup copy of the critical data from the SNT (next available document ID, next surface ID, next background job ID, and encoded date/time stamp):

```
5018595 3344 98 856828856
```

The system updates the snt.chkpt file any time it allocates a new surface ID or creates a background job, once for every 1,000 new documents allocated. The system time stamps the file to help you determine how old the checkpoint file is compared to the last SNT restore. It decodes the encoded timestamp for display when you start SNT_update. (See the example on **page 391**.)

Automatic Checkpoint Verification

Each time Document Services starts up during a start or restart of FileNet software, the system compares the values in the snt.chkpt file against the values in the SNT of the permanent database. If any of the

snt.chkpt file values is higher than those in the SNT, Document Services terminates and logs the following message:

```
Severe Error condition: The Scalar Numbers Table is behind the snt.chkpt
file. This should only happen after a Permanent DB restore has been done.
Continuing with this condition may cause multiple documents to be
committed with the same doc ID. To resolve this problem, you must either
remove the snt.chkpt file (if its contents are invalid) or update the
Scalar Numbers Table with the SNT_update program. Doc Services will not
function until this problem is resolved.
```

SNT_update reads the contents of the snt.chkpt file and adds 1,000 to the next available document ID value to ensure that the scalar numbers table is updated to the highest possible value. (Remember that the snt.chkpt file is only updated for every 1,000 newly allocated documents.) For example, if the value of the next document ID in the snt.chkpt file is 5018595, SNT_update modifies the value in the checkpoint file to 5019595 during the update.

Use

The SNT_update program updates the scalar numbers table of the MKF permanent database from information obtained from the SNT checkpoint file. You could use SNT_update to prevent creation of duplicate documents, surfaces, and background jobs under one of the following conditions:

- After restoring the MKF permanent database from tape: You might restore data if you lose the database due to a magnetic media crash and the /fnsw/local (\fnsw_loc) partition is still available and undamaged.

- After initializing the software: You might use an initialization tool (such as ds_init, fn_util init, or fn_util initperm) to set software parameters back to their original values.

Syntax

SNT_update

The program prompts you to quit or continue before it begins updating the scaler numbers table (SNT).

Checklist

Before you use SNT_update, be aware of the following:

- You must not run SNT_update if the /fnsw/local (\fnsw_loc) partition has been lost or damaged. If the partition is lost or damaged, your service representative for assistance in restoring a valid SNT from a backup copy of the snt.chkpt file.
- SNT_update may waste up to 1,000 document IDs but does not waste any surface IDs or background job numbers.

Procedure

The following procedure updates the scalar numbers table with a valid checkpoint file.

Tip In the following procedure, references to the directory path for the snt.chkpt file depend on your platform, as follows:

For UNIX platforms: /fnsw/local/sd/snt.chkpt
For Windows Server platforms: \fnsw_loc\sd\snt.chkpt

1. Before you start the restore or initialization operation, make a copy of the magnetic disk-resident scalar numbers table checkpoint file, snt.chkpt.

   You can copy the checkpoint file to tape or to a directory such as /fnsw/local/tmp (in UNIX) or \fnsw_loc\tmp (in Windows Server).

Note If you cannot save the checkpoint file prior to performing a restore or initialization or if a magnetic disk crash has corrupted the checkpoint file, contact your service representative immediately.

2. Perform the restore or initialization operation.

3. Verify that the restore or initialization operation did not overwrite the disk-resident snt.chkpt file.

   If snt.chkpt was overwritten, copy the **saved** version of the snt.chkpt file (created in step 1 above) back into /fnsw/local/sd or \fnsw_loc\sd, depending on your platform.

4. Restart the FileNet software.

   The Document Services initialization routine automatically compares the snt.chkpt file with the SNT table and, if necessary, issues messages directing you to run SNT_update.

5. Enter **SNT_update** at the command line.

An example of the confirmation display is shown below. The values for surface, job, and document IDs are for illustration only. The values in your display will be different:

```
corona>SNT_update
"The time stamp on your checkpoint is Tue Feb 26 13:45:52 2002
"The SCALAR_NUMBERS table values in the Permanent Database are:
                Next Surface  ID (MKF) = 3344
                Next Job      ID (MKF) = 98
                Next Document ID (MKF) = 5018595
The check point file values are:
                Next Surface  ID (checkpoint) = 3344
                Next Job      ID (checkpoint) = 98
                Next Document ID (checkpoint) = 5018598

    ===> New Next Document ID will be = 5019598

Do you wish to update the Permanent Database Scalar numbers with the
checkpoint
values? (y/n):
```

6 Respond to the update prompt.

If you reply **n**, SNT_update terminates.

If you reply **y**, one of the following messages displays indicating successful completion or errors that prevent the successful update of the scalar numbers table:

- “Scalar numbers table updated”
- “Update was not necessary
  Scalar numbers table already up to date”
- “Scalar numbers table update failed, err=<err_code>”

where <err_code> is a value indicating the type of error encountered. SNT_update terminates after displaying this message.

If SNT_update does not complete successfully, contact your service representative for assistance.

Appendix I – Sample Scripts and Backus-Naur Form Scripts

This appendix describes the sample scripts provided with the FileNet software. You can use these scripts as the basis for creating your own unique scripts. In addition, the **"Backus-Naur Form Script Specification" on page 399** is included to assist programmers in developing script language programs. You can also use the EBR_genscript tool (see **"EBR_genscript" on page 268**) to create customized scripts.

Sample scripts are located in the following directories:

UNIX

/fnsw/local/EBR/samples For UNIX systems

WIN

\fnsw_loc\EBR\samples For Windows Server systems
or any directory of your choice.

A special READ_ME file describes each script and file in the directory.

Each sample backup and restore script in the EBR samples directory includes the required statements to perform a backup or restore. For your convenience, comments point out required statements and parameters, special syntax considerations, and other descriptive information.

The samples directory also contains sample include files and template files that contain backup options and restore options. Template files are described in **Chapter 5, "Developing Your Scripts," on page 147**.

Note New sample scripts may be added and existing sample scripts may be modified in subsequent versions. The scripts are copied into the FileNet directories listed above, replacing the existing scripts. Therefore, the scripts you modify and use should be in your own separate directory. Create a directory and copy all FileNet-provided scripts into your directory. Edit and test the script copies in your directory.

Protect your customized scripts by backing them up to tape with a file copy utility. You may also want to have a printed copy available.

Sample Script Descriptions

FileNet-provided sample scripts for backup and restore are the following:

- **full_off.bac** script performs a full, offline backup of the security, permanent, transient, and index databases and cache. Two threads are used for the backup. The script references include files that contain the dataset definitions.
- **full_off.res** is the corresponding restore script to full_off.bac. The full_off.res script performs a restore of the security, permanent, transient, and index databases and cache from full, offline backup tapes. The interval_restore_follows option is set to false so that EBR performs only a full restore.
- **full_on.bac** script performs a full, online backup of the security, permanent, and index databases using two threads.
- **full_on.res** is the corresponding restore script to full_on.bac. This script performs a restore of the security, permanent, and index databases from full, online backup tapes. The interval_restore_follows option is set to false so that EBR performs only a full restore.

- **ival_on.bac** script performs an interval, online backup of the permanent, security, and index databases using one thread.
- **ival_on.res** is the corresponding restore script to ival_on.bac. The inval_on.res script performs a restore of the permanent, security, and index databases from an interval, online backup tape.
- **cache.bac** script performs a full, offline backup of both the transient database and cache using one thread. The transient database and cache must always be backed up and restored together.
- **cache.res** is the corresponding restore script to cache.bac. The cache.res script performs a restore of the transient database and cache from full offline backup tapes.

The samples directory contains other scripts that provide examples of shell commands that perform a syntax check on an EBR sample script. Others provide samples for scheduling a backup. For example, use the **cron.sh** script to schedule an unattended backup using crontab or the AT command. In a Windows Server environment, use the **run_bkup.sh** script to run daily backups using the AT command.

Include File Descriptions

A sample include file, datasets.inc, located in /fnsw/local/EBR/samples (UNIX) or \fnsw_loc\EBR\samples (Windows Server), contains the DATASETS description referenced by some EBR sample scripts. The contents of the datasets.inc file (in UNIX format) is shown below:

```
-- datasets.inc
-- Purpose:
--    This include file contains the DATASETS description referenced by
--    EBR backup/restore sample scripts.
--
--    Signature file directory for Oracle database should exist before
--    running scripts.

DATASETS

    sec: MKF
      location = $disk_location;
      base_data_file = "/fnsw/dev/1/sec_db0";
    end_MKF

    perm: MKF
      location = $disk_location;
      base_data_file = "/fnsw/dev/1/permanent_db0";
    end_MKF

(continued on next page)
```

```
(continued from previous page)

    tran: MKF
      location = $disk_location;
      base_data_file = "/fnsw/dev/1/transient_db0";
      transient_db;
    end_MKF

    cache: cache
      location = $disk_location;
      transient_db = tran;
      permanent_db = perm;
      security_db = sec;
    end_cache

   inxdb: Oracle
      location = $disk_location;
      signature_file_directory = "/fnsw/local/tmp/ora_sig";   --user
defined directory
    end_Oracle

    order_constraints
      sec, perm, tran before cache;
    end_order_constraints

END_DATASETS
```

Backus-Naur Form Script Specification

EBR script language is based on the Backus-Naur Form script specification. Programmers responsible for the script development can use the following Backus-Naur Form specification structure when creating new EBR scripts or editing EBR sample scripts.

```
<EBR_script> ::=
        <backup_script> |
        <restore_script>

<null> ::=

<backup_script> ::=
        <syntax_format_level>
        <backup_global_parameters_section>
        <dataset_definitions_section>
        <backup_options_section>
        <threads_section>

<syntax_format_level> ::=
        EBR_script ( format_level = 2 ; ) ;

<backup_global_parameters_section> ::=
        BACKUP_GLOBAL_PARAMETERS
            volume_group = <volume_group_name> ;
            expiration = <integer> <expiration_unit> ;
            <delayed_start_spec>
            <tape_mount_timeout_spec>
        END_BACKUP_GLOBAL_PARAMETERS

<volume_group_name> ::=
        <upper_case_letter> |
        <volume_group_name> <upper_case_letter> |
        <volume_group_name> _ |
        <volume_group_name> <decimal_digit>
        -- limit of 28 characters total
```

```
<expiration_unit> ::=
        day | days | week | weeks | month | months
<delayed_start_spec> ::=
        <null> |
        start_date = <integer_year> / <integer_month> / <integer_day> ;
        start_time = <integer_hour> : <integer_minute> ;
<tape_mount_timeout_spec> ::=
        <null> |
        tape_mount_timeout = <integer_second>;
<integer_year> ::=
        <integer>
        -- must be exactly four digits

<integer_month> ::=
        <integer>
        -- must be from 1 to 12

<integer_day> ::=
        <integer>
        -- must be from 1 to 31

<integer_hour> ::=
        <integer>
        -- must be from 0 to 23

<integer_minute> ::=
        <integer>
        -- must be from 0 to 59

<integer_second> ::=
        <integer>

<dataset_definitions_section> ::=
        DATASETS
            <dataset_definition_list>
            <order_constraints_section>
```

```
        END_DATASETS

<dataset_definition_list> ::=
        <dataset_definition_subsection> |
        <dataset_defintion_list> <dataset_definition_subsection>

<dataset_definition_subsection> ::=
        <MKF_database_definition> |
        <Oracle_database_definition_subsection> |
        <cache_definition_subsection> |
        <raw_partition_defintion>

<MKF_database_definition> ::=
        <dataset_id> : MKF
            <host_location>
            base_data_file = "<full_path_name>" ;
            <tran_db_flag>
        end_MKF
<tran_db_flag> ::=
        <null> |
        transient_db ;

<dataset_id> ::=
        <identifier>

<host_location> ::=
        location = "<host_location>" ;

<Oracle_database_definition_subsection> ::=
        <dataset_id> : Oracle
            <host_location>
            signature_file_directory = "<full_path_name>" ;
        <parameter_file_location>
        end_Oracle

<parameter_file_location> ::=
        null |
```

```
        parameter_file = "<full_path_name>";

<cache_definition_subsection> ::=
        <dataset_id> : cache
            <host_location>
            transient_db = <dataset_id> ;
            permanent_db = <dataset_id> ;
            security_db = <dataset_id> ;
        end_cache

<raw_partition_definition_subsection> ::=
        <dataset_id> : partition
            <host_location>
            filename = "<full_path_name>";
            <sub_partition_spec>
        end_partition

<sub_partition_spec> ::=
        <start_block_spec>
        <block_size_spec>
        <num_blocks_spec>

<start_block_spec> ::=
        <null> |
        start_block = <integer> ;

<block_size_spec> ::=
        <null> |
        block_size = <sector_size> ;
<sector_size> ::=
        512 |
        1024 |
        2048 |
        4096

<num_blocks_spec> ::=
        <null> |
```

```
        num_blocks = <integer> ;

<order_constraints_section> ::=
        order_constraints
            <order_constraint_list>
        end_order_constraints

<order_constraint_list> ::=
        <null> |
        <order_constraint> |
        <order_constraint_list> <order_constraint>

<order_constraint> ::=
        <before_list> before <dataset_id> ;

<before_list> ::=
        <dataset_id> |
        <before_list> , <dataset_id>

<backup_options_section> ::=
        BACKUP_OPTIONS
            <backup_option_list>
        END_BACKUP_OPTIONS
<backup_option_list> ::=
        <null> |
        <backup_option_subsection> |
        <backup_option_list> <backup_option_subsection>
<backup_option_subsection> ::=
        <MKF_backup_option_subsection> |
        <Oracle_backup_option_subsection> |
        <cache_backup_option_subsection> |
        <partition_backup_option_subsection>

<MKF_backup_option_subsection> ::=
        <dataset_id> : backup_options
            { full_backup | interval_backup } ;
            { online_backup | offline_backup } ;
```

```
        end_backup_options

<Oracle_backup_option_subsection> ::=
        <dataset_id> : backup_options
            { full_backup | interval_backup } ;
            { online_backup | offline_backup } ;
            archive_redo_log_retention = <integer> days ;
        end_backup_options

<cache_backup_option_subsection> ::=
        <dataset_id> : backup_options
            full_backup;
        end_backup_options

<partition_backup_option_subsection> ::=
        <null> |
        <dataset_id> : backup_options
        end_backup_options

<device_specification_section>::=
        DEVICE_SPECIFICATIONS
            <device_specification_list>
        END_DEVICE_SPECIFICATIONS

<device_specification_list>::=
        <device_specification_subsection>|
        <device_specification_list> <device_specification_subsection>

<device_specification_subsection>::=
         <disk_file_device_specification>| <tape_device_specification> |
         <tape_library_device_specification>

<disk_file_device_specification>::=
         <device_id>:disk_file;
          location = "<host_location>";
          filename = "<full_path_name>";
         end_disk_file
```

```
<tape_device_specification>::=
         <device_id>:tape drive;
          location = "<host_location>";
          drive_name_rewind = "<full_path_name>";
          drive_name_no_rewind = "<full_path_name>";
          drive_type = {8mm |4mm |DAT |QIC};
         end_tape_drive

<tape_library_device_specification>::=
         <device_id>:tape_library;
          location = "<host_location>";
          library_device = "<full_path_name>";
          library_type = {Exabyte |"EXB-210" |"EXB-218" |"EXB-220"};
          <tape_library_reservation>

          tape_drives
           num_drives = <num_drives>;
           <tape_library_tape_drive_list>
          end_tape_drives
         end_tape_library

<tape_library_reservation>::=
         null |
         reserve_slots = <reservation_list>;
         reserve_drives = <reservation_list>;

<reservation_list>::=
         <integer> |
         <integer>, <reservation_list>

<num_drives>::=
         <integer>
<tape_library_tape_drive_list>::=
         <tape_library_tape_drive_subsection> |
        <tape_library_tape_drive_list> <tape_library_tape_drive_subsection>
end_tape_drives
```

```
<tape_library_tape_drive_subsection>::=
          drive <drive_id>
           drive_name_rewind = "<full_path_name>"";
           drive_name_no_rewind = "<full_path_name>"0";
           drive_type = {8mm |4mm |DAT};

<drives_id>::=
          <integer>

<thread_section>::=
          THREAD
           num_threads = <num_threads>;
           <thread_list>
          END_THREADS

<num_threads>::=
          <integer>

<thread_list>::=
          <thread_subsection> |
          <thread_list> <thread_subsection>

<thread_subsection>::=
          thread <thread_number>
           device = <device_subsection>;
           volume_serial = <volume_serial_name>;
           <dataset_list>
          end_thread

<thread_number>::=
          <integer>
          -- thread numbers are from 1 to <num_threads>, and must
          -- occur in increasing numberical order.

<volume_serial_name>::=
         <upper_case_letter> |
```

```
        <volume_group_name> <upper_case_letter> |
        <volume_group_name>_ |
        <volume_group_name> <decimal_digit>
        -- limit of 12 characters total

<dataset_list>::=
        datasets <dataset_stripe_list>;

<dataset_stripe_list>::=
        <dataset_stripe> |
        <dataset_stripe_list> , <dataset_stripe>
<dataset_stripe>::=
        <dataset_id> |
        <dataset_id> (part <part_num> of <num_parts> )
        -- <part_num> is from 1 to <num_parts>.
        -- All parts from 1 to <num_parts> must be mentioned exactly once
        -- in the <dataset_list>'s in the script.
        -- Each occurence of the dataset in a <dataset_list> must have
        -- the same value for <num_parts>.
        -- A dataset may be mentioned at most once in a <dataset_list>.

<part_num>::=
        <integer>

<num_parts>::=
        <integer>

<device_subsection>::=
         <tape_device_subsection>|
         <disk_file_device_subsection> |
         <tape_library_device_subsection>

<tape_device_subsection>::=
         <device_id>

<disk_file_device_subsection>::=
         <device_id>
```

```
<tape_library_device_subsection>::=
         <device_id> (drive <drive_number>) |
         <device_id> (drive <drive_number>, slot <slot_number>);

<drive_number>::=
         <integer>

<slot_number>::=
         <integer>

<restore_script> ::=
        <syntax_format_level>
        <restore_global_parameters_section>
        <dataset_definitions_section>
        <restore_options_section>
        <threads_section>

<restore_global_parameters_section> ::=
        RESTORE_GLOBAL_PARAMETERS
            volume_group = <volume_group_name>
            <tape_mount_timeout_spec>
        END_RESTORE_GLOBAL_PARAMETERS

<restore_options_section> ::=
        RESTORE_OPTIONS
            <restore_option_list>
        END_RESTORE_OPTIONS

<restore_option_list> ::=
        <null> |
        <restore_option_subsection> |
        <restore_option_list> <restore_option_subsection>
<restore_option_subsection> ::=
        <MKF_restore_option_subsection> |
        <Oracle_restore_option_subsection> |
        <cache_restore_option_subsection> |
```

```
        <partition_restore_option_subsection> |
<MKF_restore_option_subsection> ::=
        <dataset_id> : restore_options
            { full_restore ; interval_restore_follows = { true | false } ; |
              interval_restore } ;
            <reconfigure_onto>
        end_restore_options

<reconfigure_onto> ::=
        <null> |
        reconfigure_onto "<full_path_name>" ;

<Oracle_restore_option_subsection> ::=
        <dataset_id> : restore_options
            { full_restore ; interval_restore_follows = { true | false } ; |
              interval_restore } ;
            <restore_control_file>
            <restore_redo_logs>
            <rollforward_options>;
        end_restore_options

<restore_control_file> ::=
        <null> |
        restore_control_file = { true | false } ;

<restore_redo_logs> ::=
        <null> |
        restore_redo_logs = { true | false } ;

<rollforward_options> ::=
        <null> |
        rollforward = { true | false } ;

<cache_restore_option_subsection> ::=
        <dataset_id> : restore_options
            full_restore ;
            interval_restore_follows = false ;
```

```
        end_restore_options

<partition_restore_option_subsection> ::=
        <null> |
        <dataset_id> : restore_options
           <restore_onto>
        end_restore_options

<restore_onto> ::=
        <null> |
        restore_onto "<full_path_name>" ;
```

Appendix J – Restoring Oracle

This appendix presents the EBR options you may need to restore Oracle databases from your backups. Depending on the reason for a restore, the restore options available to you differ. For some restore cases, you use EBR. For others, you use Oracle procedures. For reference, see the discussion under **"Help for Unusual Circumstances" on page 227**.

This appendix also presents information about restoring Oracle datasets to a different system.

Tip If you choose to retain older archive redo logs for disaster recovery purposes, you must copy them to tape manually. EBR does not back up the archive redo logs located in the directory you specified in the log_archive_dest parameter because a successful full backup and regularly scheduled interval backups pick up the changes recorded in the archive logs. Older archive redo logs are only useful in the highly unlikely event that a triple failure occurs: the disk containing your Oracle database crashes, your last full backup tape is lost or destroyed, and the magnetic disk containing your Oracle archive redo log crashes.

You can prevent this triple failure scenario by placing your archive redo log on a different spindle than your database. If you place them on the same spindle, consider manually backing up older archive redo logs to tape.

The tables in this appendix describe the options necessary to restore full and interval offline and online backups of the Oracle index database.

Options to Restore a Full Offline Index Database Backup

A full offline backup of the Oracle index database includes the data file, control file, and online redo logs. (EBR automatically backs up the magnetic disk-resident archive redo log.) The table below describes the restore options to specify for a given failure situation that requires restoring one or more of the backed up database components.

Note Options for an interval restore are the same as for full restore, interval_restore_follows=false

The column headings of the table describe the type of restore you want to perform. Each table row identifies a type of failure and the restore options you should specify to restore the failed database component. Where appropriate, brief explanations of the option results are given.

Options to Restore a Full Offline Oracle Index Database Backup

Options for full restore, interval_restore_follows = false	Options for full restore, interval_restore_follows = true
Failure1. When only the data file is bad, specify: **restore_control_file=false** **restore_redo_logs=false** Restores data file and rolls the database forward to the last transaction. Since the default is false, specifying this is optional.	Failure 1. When only the data file is bad, specify: **restore_control_file=false** **restore_redo_logs=false** No rollforward occurs. Since the default is false, specifying this is optional.

Options to Restore a Full Offline Oracle Index Database Backup, Continued

Options for full restore, interval_restore_follows = false	Options for full restore, interval_restore_follows = true
Failure 2. When data file and control file are bad, specify: **restore_control_file=true** **restore_redo_logs=false** Restores the data file and control file then rolls the database forward to the last transaction. However, you must **manually** reset the archive redo log with the following Oracle SQL*Plus command: **alter database open resetlogs;** Also **manually** remove any archived redo logs on magnetic disk.	Failure 2. When data file and control file are bad, specify: **restore_control_file=true** **restore_redo_logs=false** No rollforward occurs.
Failure 3. When data file, control file, and redo logs are bad, specify: **restore_control_file=true** **restore_redo_logs=true** Same as failure 2 above. However, EBR cannot guarantee that the database will roll forward to the last transaction. EBR issues a warning at the end of the job.	Failure 3. When data file, control file, and redo logs are bad, specify: **restore_control_file=true** **restore_redo_logs=false** No rollforward occurs.

Options to Restore a Full Offline Oracle Index Database Backup, Continued

Options for full restore, interval_restore_follows = false	Options for full restore, interval_restore_follows = true
Failure 4. When only redo logs are bad, refer to the *Oracle Server Administrator's Guide* or *Oracle Server Backup and Recovery Guide* for procedures on how to recover from corrupt online redo logs.	Failure 4. When only redo logs are bad, refer to the *Oracle Server Administrator's Guide* or *Oracle Server Backup and Recovery Guide* for procedures on how to recover from corrupt online redo logs.
Failure 5. When only the control file is bad, **do not run EBR**. Run Oracle commands to recreate the file.	

Note If you initialized the Oracle database before the restore, you must restore the control files, data file, and redo logs from a full, offline Oracle backup. Follow the procedures as described in **"Restoring Oracle Datasets to a Different System" on page 421**.

Options to Restore a Full Online Index Database Backup

A full online backup of the Oracle index database includes the data file and magnetic disk-resident archive redo log file. EBR does **not** back up the online redo logs. The table describes the restore options to specify for a given failure situation that requires restoring one or more of the backed up database components.

Note Options for an interval restore are the same as for full restore, interval_restore_follows=false.

The column headings of the table describe the type of restore you want to perform. Each table row identifies a type of failure and the restore

options you should specify to restore the failed database component. Where appropriate, brief explanations of the option results are given.

Options to Restore a Full Online Oracle Index Database Backup

Options for full restore, interval_restore_follows = false	Options for full restore, interval_restore_follows = true
Failure 1. When only the data file is bad, specify: **restore_control_file=false** **restore_redo_logs=false** Restores data file and rolls the database forward to the last transaction. Since the default is false, specifying this is optional.	Failure 1. When only the data file is bad, specify: **restore_control_file=false** **restore_redo_logs=false** Partial recovery. (See **"restore_control_file Option" on page 228** and **"restore_redo_logs Option" on page 230** for details.) Since the default is false, specifying this is optional.
Failure 2. When data file and control file are bad, specify: **restore_control_file=true** **restore_redo_logs=false** Restores the data file and control file, then rolls the database forward to the last transaction. However, you must **manually** reset the archive redo log with the Oracle Server Manager command: **alter database open resetlogs;** Also **manually** remove any archived redo logs on magnetic disk.	Failure 2. When data file and control file are bad, specify: **restore_control_file=true** **restore_redo_logs=false** Partial recovery. (See **"restore_control_file Option" on page 228** and **"restore_redo_logs Option" on page 230** for details.)

Options to Restore a Full Online Oracle Index Database Backup, Continued

Options for full restore, interval_restore_follows = false	Options for full restore, interval_restore_follows = true
Failure 3. When data file, control file, and redo logs are bad, specify: **restore_control_file=true** Restores the data file and control file, then attempts to roll the database forward to the last transaction. EBR ignores the restore_redo_logs option and displays a warning message at the end of the job. Redo log backups are not contained in a full online Oracle backup. First back up any archived redo logs on magnetic disk before **manually** removing them. **Then manually** reset the archive redo log with the Oracle Server Manager command: **alter database open resetlogs;**	Failure 3. When data file, control file, and redo logs are bad, specify: **restore_control_file=true** **restore_redo_logs=true** Partial recovery. (See **"restore_control_file Option" on page 228** and **"restore_redo_logs Option" on page 230** for details.)
Failure 4. When only redo logs are bad, refer to the *Oracle Server Administrator's Guide* or *Oracle Server Backup and Recovery Guide* for procedures on how to recover from corrupt online redo logs.	Failure 4. When only redo logs are bad, refer to the *Oracle Server Administrator's Guide* or *Oracle Server Backup and Recovery Guide* for procedures on how to recover from corrupt online redo logs.

Options to Restore an Interval Index Database Backup

The table describes the restore options you may perform when you have restored a full offline or online Oracle database backup and you follow with a restore from an interval offline or online Oracle database backup.

An interval offline backup of the Oracle index database contains the data file, control file, and online redo logs. EBR automatically backs up the magnetic disk-resident archive redo log.

An interval online backup of the Oracle index database contains the data file and magnetic disk-resident archive redo log file. EBR does not back up the online redo logs.

Note If you have initialized the Oracle database before the restore, you must restore the control files, data file, and redo logs from a full **offline** Oracle backup.

Follow this sequence if you plan to initialize the Oracle database before a full online restore followed by an interval online restore:

- full offline backup
- full online backup
- interval online backup (if database has changed since online backup)
- initialize Oracle database
- full offline restore
- full online restore
- interval restore
- manual recovery from archiving files

(See **"Manual Recovery" on page 420** for information on this process.

The column heading of the following table describes the type of restore you perform. Each table row identifies a type of failure and the restore options you should specify to restore the failed database component. Where appropriate, brief explanations of the option results are given.

Options to Restore an Interval Offline or Online Oracle Index Database Backup

Options for an interval offline restore, following a full offline/online restore	Options for an interval online restore, following a full offline/online restore
Failure1. When only the data file is bad, specify: **restore_control_file=false** **restore_redo_logs=false** Restores data file and rolls the database forward to the last transaction. Since the default is false, specifying this is optional.	Failure 1. When only the data file is bad, specify: **restore_control_file=false** **restore_redo_logs=false** **rollforward=false** Restores the data file but does not roll the database forward to the last transaction. After running EBR to restore the database, you must manually recover the database using the archive file(s) and the current redo log (if applicable). See **"Manual Recovery" on page 420**.
Failure 2. When data file and control file are bad, specify: **restore_control_file=true** **restore_redo_logs=false** Restores the data file and control file then rolls the database forward to the last transaction. However, you must **manually** reset the archive redo log with the Oracle Server Manager command: **alter database open resetlogs;** Also **manually** remove any archived redo logs on magnetic disk.	Failure 2. When data file and control file are bad, specify: **restore_control_file=false** **restore_redo_logs=false** **rollforward=false** Restores the data file and control file, but does not roll the database forward to the last transaction. After running EBR to restore the database, you must manually recover the database using the archive file(s). See **"Manual Recovery" on page 420**.

Options to Restore an Interval Offline or Online Oracle Index Database Backup, Continued

Options for an interval offline restore, following a full offline/online restore	Options for an interval online restore, following a full offline/online restore
Failure 3. When data file, control file, and redo logs are bad, specify: **restore_control_file=true** **restore_redo_logs=true** Same as failure 2 above. However, EBR cannot guarantee that the database will roll forward to the last transaction. EBR issues a warning at the end of the job.	Failure 3. When data file, control file, and redo logs are bad, specify: **restore_control_file=true** **rollforward=false** Restores the data file and control file, but does not attempt to roll the database forward to the last transaction. EBR ignores the restore_redo_logs option and displays a warning message at the end of the job. Redo log backups are not contained in an interval online Oracle backup. After running EBR to restore the database, you must manually recover the database using the archive file(s). See **"Manual Recovery" on page 420**. Note: Use the default value for rollforward in a full online restore. (Don't use this option in the full restore script if it's going to be followed by an interval restore.)
Failure 4. When only redo logs are bad, refer to the *Oracle Server Administrator's Guide* or *Oracle Server Backup and Recovery Guide* for procedures on how to recover from corrupt online redo logs.	Failure 4. When only redo logs are bad, refer to the *Oracle Server Administrator's Guide* or *Oracle Server Backup and Recovery Guide* for procedures on how to recover from corrupt online redo logs.
Failure 5. When only the control file is bad, **do not run EBR**. Run Oracle commands to recreate the file.	

Manual Recovery

To recover the Oracle database manually, follow these steps:

1. Look at the Oracle initialization file (e.g., init.ora) to find out the value of log_archive_dest, the location of the archive logs.

2. Locate the archive log directory and list its contents. Note the files that exist there.

3. As any member of the **dba** group, such as **oracle** user or **fnsw** user, start and mount the Oracle database, and begin the recovery by entering SQL*Plus commands similar to these:

   sqlplus "/ as sysdba"

   (The double-quotes are required; the space between **/** and **as** is optional.)

   SQL> **startup pfile=/fnsw/local/oracle/init.ora mount**
   SQL> **recover using backup controlfile until cancel;**

4. The SQL*Plus displays a list of options. After you've made sure the listed archive file exists (as listed in the log_archive_dest directory), choose the default (RET*) option for recovery.

5. Repeat the recovery of each archive log in the log_archive_dest directory.

   - After the last archive log has been recovered into the database, recover the redo logs (if applicable).

   - After all archive and redo logs have been recovered, enter **cancel**.

6 When you have finished recovering the archive and redo logs, enter the following command:

SQL> **alter database open resetlogs;**

7 After the Oracle database has started up, shut it down normally:

SQL> **shutdown normal**

8 Exit from the SQL*Plus by entering:

SQL> **exit**

9 Remove the old log files and backup(s) from the server.

10 Perform a full offline backup to reset the log files.

11 At this point, the Oracle database is fully recovered and can be put back into normal production.

Restoring Oracle Datasets to a Different System

Occasionally you may need to restore an Oracle dataset to another system. The following information is important to understand before you perform such a procedure.

An Oracle dataset backup that you restore to a different system must be a full, offline backup. Do not attempt to restore an online Oracle backup to a different system because an online Oracle backup does not contain backups of the online Oracle redo logs. Unless the Oracle control files, data file, and online redo logs are synchronized with each other, your Oracle database cannot start up. Refer to your *Oracle System Administrator's Guide* for more information on database table recovery and redo logs.

The Oracle configuration should be identical for both the source system (the system that contains the original Oracle dataset) and the target system (the system to which you are restoring the dataset).

The highest-numbered configuration database (CDB) file reflects the current configuration of your FileNet system. The format of the configuration database file is:

/fnsw/local/sd/conf_db/IMS_*nnn*.cdbfor UNIX platforms

<drive>:\fnsw_loc\sd\conf_db\IMS_*nnn*.cdbfor Windows Server platforms

where *nnn* is the file number.

Check the current CDB file of the target and source systems to verify that the Oracle partitions are configured exactly the same on each.

If archive log mode is enabled on the source system, enable archive log mode on the target system before you begin the restore. Create the archive log destination directory on the target system. The name of the archive log directory on the target system must be the same as the archive log directory name on the source system. Remove existing archive log files from the target system before you start the restore.

Your restore script should contain the following two options for the Oracle dataset:

```
restore_control_file=true;
restore_redo_logs=true;
```

Use the following procedure if you need to restore an Oracle dataset to a different system.

1 Collect the backup tapes from a full, **offline** backup of the Oracle dataset from the source system.

CAUTION Do not attempt to restore an **online** Oracle backup to a different system because an online Oracle backup does not contain backups of the online Oracle redo logs. Unless the Oracle control files, data file, and online redo logs are synchronized with each other, your Oracle database cannot start up.

2 Verify that the configuration database (CDB) files of the target and source systems are identical.

The Oracle configuration **must** be identical for both the target system and the source system. The CDB files of the target and source systems should have identical configurations for the Oracle partitions (such as oracle_db0 and oracle_rl0). However, the sizes of the partitions on the target system can be larger than those of the source system.

3 Verify that the setting of archive log mode is the same on both the target and source systems.

If archive log mode is enabled on the source system, enable archive log mode on the target system using the same archive log directory name on the target system as you used on the source system. (See **“Appendix F – Enabling Archive Log Mode” on page 377** for a detailed procedure.) Then remove any existing archive log files from the target system.

4 Establish the appropriate restore mode on the target system.

See **“Restore Procedure” on page 253**.

5 Run your restore script on the target system.

Your restore script should contain the following two options for the Oracle dataset:

```
restore_control_file=true;
restore_redo_logs=true;
```

For additional information on setting options for your restore scripts, see **Chapter 5, “Developing Your Scripts,” on page 147**.

Appendix K – Calculating Throughput

From information in the EBR progress log, you can compute throughput of the disk, the network, and the tape drive. You can also determine how much disk data was read and how much compressed tape data was sent over the network and to the tape drive. (Less data is sent over the network than is sent to the tape drive.)

Information from the following progress log sample is used to describe a method you can use as a guide for calculating throughput for your system. Use the information in conjunction with the disk, tape, and network throughput examples that start on **page 428**. The progress log is abbreviated as noted with ellipses:

```
Enterprise Backup/Restore Progress Log
Mon Jan  13 15:31:16 2002
Processes and their assigned letters:
M = master process. Makes RPC's to all tape rqh. processes (one per
thread).
T = tape requesthandler process.  Forks a "t" and an "n" process.
    t = tape worker process.   Performs I/O on a single tape drive.
    n = network (N/W) disk client process.  Makes RPC's to a disk rqh.
process.
D = disk requesthandler process.  Forks a "d" and a "c" process.
    d = disk worker process.  Performs I/O on files of a disk dataset.
    c = disk data compressor/decompressor process.

(continued on next page)
```

```
(continued from previous page)

(In the next line, 'N' is thread Number, 'L' is process Letter:)
hh:mm:ss NL   pid    message
-------- -- -------- ------------------------------------------------
15:31:16 0M    39755 Master process:  BEGIN BACKUP
15:31:18 1n    32366 Begin N/W disk client process for backup
15:31:18 1n    32366 Begin backup of dataset or1 (part 1 of 1)
15:31:18 1t    40273 Begin tape worker process for backup. Backup to disk
file
                     "/tmp/kenr/diskdata.bu1" at "rojo" .
15:31:18 1t    40273 NEED TAPE serial=R1 volume_group=VG1 IN DRIVE
                     /tmp/kenr/diskdata.bu1 AT rojo !
15:31:18 1t    40273 backup disk file opened
15:31:18 1t    40273 Begin backup of dataset or1 (part 1 of 1)
15:31:19 1d    38277 Begin disk worker process, thread 1 of 1, at
                    "TapeServer1:rojo:FileNet”
15:31:19 1d    38277 Begin backup part 1 of 1 of dataset “or1”
15:31:37 1n    32336 Got 1 compressed blocks (32K) cumulative from disk
server
15:31:37 1d    38277 Read 2,048 bytes cumulative: “/fnsw/dev/1/oracle_db0”
...
15:31:40 1t    40273 Wrote 5 compressed blocks (32K bytes) cumulative to
tape.
15:31:42 1n    32336 Got 34 compressed blocks (32K) cumulative from disk
server.
15:32:13 1d    38227 Read 113,903,616 bytes cumulative: /fnsw/dev/1/
oracle_db0

(continued on next page)
```

```
(continued from previous page)

15:32:17 1n    32336 Got 80 compressed blocks (32K) cumulative from disk
server.
15:32:17 1d    38227 Read 129,107,968 bytes cumulative: "/fnsw/dev/1/
oracle_db0"
...
15:32:38 1d    38227 Read 209,711,104 bytes final total:
                      "/fnsw/dev/1/oracle_db0". Backup finished.
15:32:41 1n    32336 Got 81 compressed blocks (32K) cumulative from disk
server.
15:32:41 1d    38227 Read 123,904 bytes cumulative: "/usr/oracle/archlog/
arch1_8.dbf"
15:32:41 1d    38227 Read 123,904 bytes final total:
                      "/usr/oracle/archlog/arch1_8.dbf". Backup finished.
15:32:41 1d    38227 Read 108,032 bytes final total:
                      "/fnsw/local/tmp/EBR/tmp/orct14br1VMC2m". Backup
finished.
15:32:41 1t    40273 Wrote 85 compressed blocks (32K bytes) cumulative to
tape.
15:32:43 1d    38227 SUCCESSFUL BACKUP of dataset or1
15:32;44 1n    32336 Begin backup of dataset t1 (part 1 of 1)
...
15:33:21 1n    32336 SUCCESSFUL end N/W disk client process for backup
15:33:21 1t    40273 Wrote 139 compressed blocks (32K) final total to
tape.
15:33:21 1t    40273 Ejecting tape at rojo and writing tape mark.
15:33:21 1t    40273 SUCCESSFUL end N/W backup tape worker process
15:33:21 0M    39755 Closing thread 1
15:33:22 0M    39755

BACKUP SUCCESSFULLY COMPLETED
```

Disk Throughput Calculation Example

To calculate the throughput of disk, select two disk worker (1d) log entries for a dataset. Usually one entry from near the beginning of the backup and one near the end of the backup is sufficient. Then select the "Backup finished" disk worker entry to get the final total of bytes read for the dataset.

In the example, we selected the following entries for the disk worker for thread 1, which backs up the Oracle dataset "or1":

15:31:19 1d	38227	Begin backup part 1 of 1 of dataset "or1"
15:31:37 1d	38277	Read 2,048 bytes cumulative: "/fnsw/dev/1/oracle_db0"
15:32:38 1d	38227	Read 209,711,104 bytes final total: "/fnsw/dev/1/oracle_db0". Backup finished.

From these entries, determine the elapsed time (15:31:37 to 15:32:38 is 61 seconds) and the cumulative number of bytes read in that time (209,711,104 – 2,048 = 209,709,056). Calculate throughput with the following formula:

Data read from disk divided by elapsed time equals throughput in bytes per second

In this example, the throughput is 3,437,853 bytes (3.28 megabytes) per second as calculated below:

209,709,056 / 61 = 3,437,853

You can use a similar method to calculate tape throughput or network throughput as shown in **"Tape Throughput Calculation Example" on page 429** and **"Network Throughput Calculation Example" on page 429**.

Tape Throughput Calculation Example

To calculate tape throughput, select two tape worker (1t) entries from the progress log in the same manner as described in **"Disk Throughput Calculation Example" on page 428**:

```
15:31:40 1t     40273     Wrote 5 compressed blocks (32K)
                           cumulative to tape.

15:32:41 1t     40273     Wrote 85 compressed blocks (32K)
                           cumulative to tape.
```

In this example, eighty 32K blocks of data (2,621,440 bytes) were written to tape in 61 seconds for a throughput of 42,974 bytes per second.

Network Throughput Calculation Example

To calculate network throughput, select two or three network worker (1n) entries from the progress log in the same manner as described in **"Disk Throughput Calculation Example" on page 428**:

```
15:31:37 1n     32336     Got 1 compressed blocks (32K)
                           cumulative from disk server.

15:31:42 1n     32336     Got 34 compressed blocks (32K)
                           cumulative from disk server.

15:32:17 1n     32336     Got 80 compressed blocks (32K)
                           cumulative from disk server.
```

In this example, seventy-nine 32K blocks of data (2,588,672 bytes) were sent over the network in 40 seconds (15:32:17 – 15:31:17 = 40 seconds) for a calculated throughput of 64,717 bytes per second.

Glossary

ageable cache

Ageable cache is time-limited storage on magnetic media. Objects that have remained in ageable cache past a specified time are eligible for deletion if space is needed to store another object. See *page cache*.

big-endian CPU

A big-endian CPU addresses the bytes of a four-byte *longword* starting from the most significant end (the “big end”) of the long-word. See *byte ordering* and *little-endian CPU*.

byte ordering

The order of bytes within a longword. See also *big-endian CPU* and *little-endian CPU*.

cache

Cache is the magnetic disk space used to store documents on the way to and from optical storage *media.* Cache can also act as permanent storage if you do not use optical storage media. Portions of cache (referred to as logical caches) are allocated for storage of different document types.

checksum

The arithmetic sum of the binary data in an object.

database
A database is a collection of logically related records or files managed by a database management subsystem. The FileNet system uses two types of databases: a third-party relational database for *index* data and WorkFlo queues and *multi-keyed file* (MKF) databases for document addresses, work in progress, and other purposes.

dataset
A collection of disk data that is backed up or restored as a unit.

DDL
Data definition language used to define MKF databases.

disk crash
A term used to describe a magnetic disk head failure.

disk mirroring
A technique for data protection in which the disks containing your data are duplicated one or more times. As data is written to the primary disk, a copy of the data is written to each of the mirrored disks. In the event of a *disk crash*, your data is still available on one or more mirrored disks.

document
Files committed to the FileNet system are documents. Documents can be images, text, forms, mixed (combinations of types), imported DOS files stored on the FileNet system's storage media, or overlays checked in by Revise users.

domain name

The domain name is the second part of the *NCH* resource name. The domain name is the system name, which is determined by you and set up by your service representative during FileNet system configuration.

drive

A drive is the physical hardware necessary for reading and writing *media*.

event logs

Event logs, created daily, contain error messages and entries for activities occurring in the system. Use the Task Manager to review event logs for FileNet software.

expiration

Expiration is a point in time after which an EBR backup tape cartridge can be reused. Prior to the expiration, the cartridge cannot be overwritten by another backup operation.

generic datasets

Non-FileNet datasets such as raw partitions and Oracle databases that are not configured by FileNet configuration tools (and thus not defined in the configuration database). Non-FileNet databases are also referred to as “site-controlled” in a database coexistence environment.

Image Services

Image Services is a set of servers and services providing a single document image *database*. The database includes a single *index database*, a single *document locator database*, and the collection of document images on storage *media*.

index database

The index database, an SQL database created and managed by an *RDBMS* such as Oracle, contains document and folder information, and can contain WorkFlo queues.

little-endian CPU

A little-endian CPU addresses the bytes of a four-byte *longword* starting from the least significant end (the “little end”) of the word. See *byte ordering* and *big-endian CPU*.

locked object

An uncommitted image or document stored in disk cache. Locked objects cannot be aged out or deleted until after committal to optical storage media.

long

See *longword*.

longword

A four-byte integer accessed as a unit by the CPU.

magnetic disk

Usually an internal hard disk on your system where the Image Services software, cache, databases, etc. are stored.

magnetic disk cache
See *cache*.

media
Media is any material on which data is stored (magnetic disk, optical disk, magnetic tape). Optical disks and MSAR disks are usually referred to as storage media in FileNet documentation.

MKF
MKF is an acronym for the FileNet Multi-Keyed File *DBMS*. The MKF subsystem manages the *transient database*, the *permanent database*, the *NCH* database, and the *security database*.

MSAR
Magnetic Storage and Retrieval provides high speed and high capacity storage libraries on magnetic disk media within the server instead of using optical media or large magnetic disk caches (Cache-only systems). For more information, refer to *MSAR Procedures and Guidelines*. To download IBM FileNet documentation from the IBM support page, see **Accessing IBM FileNet Documentation**.

NCH
The NCH (network clearinghouse) is an *MKF* database that keeps track of resources and their addresses. A *resource (*such as a user ID or a printer) is identified by a three-part name stored in the NCH database. See *object name*, *domain name*, *organization name*.

network clearinghouse
See *NCH*.

no-rewind tape device

A tape device that does not rewind when the tape is closed at the end of an operation. To rewind the tape, you must issue operating system commands to force the tape to rewind. When defining your backup device in an EBR script, you must specify both rewind and no-rewind special file names for the device. The special file names for tape devices are unique to each operating system. See your operating system documentation for special device file names for your system.

organization name

The organization name is the third part of the *NCH* resource name. Xerox Corporation registers organization names so that NCH names are globally unique.

OSAR

OSAR is an acronym for FileNet Optical Storage and Retrieval unit. See *storage library server* for a definition of OSAR server functions.

overflow reel

The system writes data to a tape until the tape overflows (writes to the end of the tape). Then another tape, called an overflow reel, is need for the application to continue writing data. EBR does not support overflow reels.

page cache

Page cache, also known as retrieval cache, is a logical *cache* containing all documents being committed to or retrieved from *storage media*. In addition, documents being retrieved from media for printing are stored in page cache before being moved to *print cache*. Page cache is an *ageable cache*.

permanent database

The permanent database stores the media location of each document entered into the system and contains tables for media surfaces, media families, and notes. See *database, MKF*.

pipelining

A high-performance technique that increases system throughput by overlapping a linear sequence of CPU and I/O activity.

RAID

An industry-standard acronym for Redundant Array of Independent Disks. RAID is a hardware and operating system software implementation of data protection through simultaneous duplication of data on multiple identical disks.

RDBMS

RDBMS is an acronym for Relational Database Management System. An RDBMS (for example, Oracle) manages the *index database* and *WorkFlo queue database*.

remote procedure call

See *RPC*.

rewind tape device

A tape device that rewinds when the tape is closed at the end of an operation. When defining your backup device in an EBR script, you must specify both rewind and no-rewind special file names for the device. The special file names for tape devices are unique to each operating system. See your operating system documentation for special device file names for your system.

root/index server

The server that checks security, locates devices, and manages the index database. On large systems, you can separate root functions (security checking and device location) and index functions from storage library functions by setting up separate servers for each function.

RPC

RPC is an acronym for remote procedure call, a technique for a process on one computer to make a request for service on another computer. A remote procedure call is the standard throughout the FileNet software for interstation communication.

script

A set of instructions, contained in an ASCII file, that describe the environment you want to back up or restore and how to perform the operation.

security database

The security database contains security information for each object (user, group, device), for the security service, for each direct membership occurrence, and for each function name and class.

server

A server is a station that provides a service in the FileNet system. Types of servers include root server, index server, batch entry server, storage library server, and application server. See also *root/index server*.

signature file

A file required to support interval backups for Oracle. The signature file (oraIDB_sig) stores a page signature for every page in the Oracle database. The page signature is used to determine which pages in the Oracle database have changed since the last full backup or the last restore.

splicing in

The process of rolling a database forward to the last complete transaction by applying data in the recovery log to a restored database. The process can only occur if no data is missing from the recovery log.

storage library

A storage library is a storage media jukebox, a unit that has a number of *slots* for containing storage media and a robotic arm that moves the media between slots, drives, and the input/output slot.

storage library server

In a system with two Image Services servers, the second server manages the *storage libraries* and includes cache storage as well as the related databases. A storage library server is sometimes referred to as an OSAR server. A system can have multiple storage library servers, each of which can manage up to four libraries. In a system with multiple storage library servers, one serves as the document locator server that keeps track of the contents of all storage libraries.

storage media

See *media*.

striping

A technique in which a single large dataset is broken into parts, called stripes. EBR backs up each stripe separately to a single tape cartridge.

synchronized

A condition in which certain FileNet datasets are updated to exactly the same point in time. This is also known as datasets being “in sync.”

thread

In EBR, a thread is a single serial stream of data to or from a tape cartridge. Except when multiple threads share a tape drive, threads of a backup process generally operate concurrently. When multiple threads share a tape drive, threads operate sequentially. The ratio of thread to tape cartridge is exactly one to one.

token

A syntax component of EBR script language. A token can be an identifier, a decimal integer, a string, or a special character.

transient database

A directory of documents, images, and available *cache* space. The transient database tracks work in progress, including the status of batches, requests to read and write media, and print request queues.

transparently remotable abstract

A shared library. A shared library has three associated files: the target abstract (or local abstract), the client stub abstract, and the server stub program. These programs allow client software to call the abstract even if the client software and the source managed by the abstract are on different hosts. The target abstract is called only on the target server.

Client software actually links to and calls the client stub abstract instead of the target abstract. By interfacing with the network clearinghouse (NCH) abstract to locate network addresses, the client stub abstract determines whether the target station is the current local station or some other (remote) station. The server stub program runs as a request handler process on the target station, deserializing requests and parameters sent by the client stub abstract. The server stub also interacts with other abstracts to establish a new network connection when necessary.

volume group

A named collection of EBR backup tape cartridges. All tapes involves in a backup or restore operation must be from the same volume group.

WorkFlo queue database

A database containing WorkFlo queue entries.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing

2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and

(ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list

of IBM trademarks is available on the Web at “Copyright and trademark information” at **www.ibm.com/legal/copytrade.shtml**.

FileNet is a registered trademark of FileNet Corporation, in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

U.S. Patents Disclosure

This product incorporates technology covered by one or more of the following patents: U.S. Patent Numbers: 6,094,505; 5,768,416; 5,625,465; 5,369,508; 5,258,855.

Index

C

N

O

P

S

U

V

W

X

Program Number: 5724-R95

Printed in USA

GC31-5548-01