UniVerse

Transaction Logging and Recovery

Version 10.3
February, 2009

IBM Corporation
555 Bailey Avenue
San Jose, CA 95141

Documentation Team: Claire Gustafson, Shelley Thompson, Anne Waite

Table of Contents

Preface

This book is for experienced UniVerse users who want to use the transaction logging subsystem to return damaged files to a state of application integrity. Users should know how to use UniVerse commands, keywords, and UniVerse BASIC statements and functions. This book is also for UniVerse administrators responsible for administering the transaction logging subsystem.

Organization of This Manual

This manual contains the following:

Chapter 1, "Introduction," is an overview of the transaction logging subsystem.

Chapter 2, "Transactions," describes transactions and transaction processing.

Chapter 3, "Transaction Logging," describes transaction logging and warmstart recovery.

Chapter 4, "The Transaction Logging Window," describes the UniVerse Admin transaction logging window.

Chapter 5, "Setting Up Transaction Logging," describes how to set up transaction logging.

Chapter 6, "Managing Transaction Logging," describes how to manage the transaction logging subsystem.

Chapter 7, "Modifying Applications," describes how to modify applications for transaction logging.

Chapter 8, "UniVerse Commands," describes UniVerse commands used with transaction logging.

Documentation Conventions

This manual uses the following conventions:

Convention	Usage
Bold	In syntax, bold indicates commands, function names, and options. In text, bold indicates keys to press, function names, menu selections, and MS-DOS commands.
UPPERCASE	In syntax, uppercase indicates UniVerse commands, keywords, and options; UniVerse BASIC statements and functions; and SQL statements and keywords. In text, uppercase also indicates UniVerse identifiers such as file names, account names, schema names, and Windows file names and paths.
Italic	In syntax, italic indicates information that you supply. In text, italic also indicates UNIX commands and options, file names, and paths.
`Courier`	Courier indicates examples of source code and system output.
`Courier Bold`	In examples, courier bold indicates characters that the user types or keys the user presses (for example, **`<Enter>`**).
[ ]	Brackets enclose optional items. Do not type the brackets unless indicated.
{ }	Braces enclose nonoptional items from which you must select at least one. Do not type the braces.
itemA | itemB	A vertical bar separating items indicates that you can choose only one item. Do not type the vertical bar.
...	Three periods indicate that more of the same type of item can optionally follow.
‰	A right arrow between menu options indicates you should choose each option in sequence. For example, "Choose File -> Exit" means you should choose File from the menu bar, then choose Exit from the File pull-down menu.

Documentation Conventions

The following conventions are also used:

- Syntax definitions and examples are indented for ease in reading.

- All punctuation marks included in the syntax—for example, commas, parentheses, or quotation marks—are required unless otherwise indicated.
- Syntax lines that do not fit on one line in this manual are continued on subsequent lines. The continuation lines are indented. When entering syntax, type the entire syntax entry, including the continuation lines, on the same input line.

UniVerse Documentation

UniVerse documentation includes the following:

UniVerse Installation Guide: Contains instructions for installing UniVerse 10.3.

UniVerse New Features Version 10.3: Describes enhancements and changes made in the UniVerse 10.3 release for all UniVerse products.

UniVerse BASIC: Contains comprehensive information about the UniVerse BASIC language. It is for experienced programmers.

UniVerse BASIC Commands Reference: Provides syntax, descriptions, and examples of all UniVerse BASIC commands and functions.

UniVerse BASIC Extensions: Describes the following extensions to UniVrese BASIC: UniVerse BASIC Socket API, Using CallHTTP, and Using WebSphere MQ with UniVerse.

UniVerse BASIC SQL Client Interface Guide: Describes how to use the BASIC SQL Client Interface (BCI), an interface to UniVerse and non-UniVerse databases from UniVerse BASIC. The BASIC SQL Client Interface uses ODBC-like function calls to execute SQL statements on local or remote database servers such as UniVerse, DB2, SYBASE, or INFORMIX. This book is for experienced SQL programmers.

Administering UniVerse: Describes tasks performed by UniVerse administrators, such as starting up and shutting down the system, system configuration and maintenance, system security, maintaining and transferring UniVerse accounts, maintaining peripherals, backing up and restoring files, and managing file and record locks, and network services. This book includes descriptions of how to use the UniAdmin program on a Windows client and how to use shell commands on UNIX systems to administer UniVerse.

Using UniAdmin: Describes the UniAdmin tool, which enables you to configure UniVerse, configure and manage servers and databases, and monitor UniVerse performance and locks.

UniVerse Transaction Logging and Recovery: Describes the UniVerse transaction logging subsystem, including both transaction and warmstart logging and recovery. This book is for system administrators.

UniVerse Security Features: Describes security features in UniVerse, including configuring SSL through UniAdmin, using SSL with the CallHttp and Socket interfaces, using SSL with UniObjects for Java, and automatic data encryption.

UniVerse System Description: Provides detailed and advanced information about UniVerse features and capabilities for experienced users. This book describes how to use UniVerse commands, work in a UniVerse environment, create a UniVerse database, and maintain UniVerse files.

UniVerse User Reference: Contains reference pages for all UniVerse commands, keywords, and user records, allowing experienced users to refer to syntax details quickly.

Guide to RetrieVe: Describes RetrieVe, the UniVerse query language that lets users select, sort, process, and display data in UniVerse files. This book is for users who are familiar with UniVerse.

Guide to ProVerb: Describes ProVerb, a UniVerse processor used by application developers to execute prestored procedures called procs. This book describes tasks such as relational data testing, arithmetic processing, and transfers to subroutines. It also includes reference pages for all ProVerb commands.

Guide to the UniVerse Editor: Describes in detail how to use the Editor, allowing users to modify UniVerse files or programs. This book also includes reference pages for all UniVerse Editor commands.

UniVerse NLS Guide: Describes how to use and manage UniVerse's National Language Support (NLS). This book is for users, programmers, and administrators.

UniVerse SQL Administration for DBAs: Describes administrative tasks typically performed by DBAs, such as maintaining database integrity and security, and creating and modifying databases. This book is for database administrators (DBAs) who are familiar with UniVerse.

UniVerse SQL User Guide: Describes how to use SQL functionality in UniVerse applications. This book is for application developers who are familiar with UniVerse.

UniVerse SQL Reference: Contains reference pages for all SQL statements and keywords, allowing experienced SQL users to refer to syntax details quickly. It includes the complete UniVerse SQL grammar in Backus Naur Form (BNF).

Related Documentation

The following documentation is also available:

UniVerse GCI Guide: Describes how to use the General Calling Interface (GCI) to call subroutines written in C, C++, or FORTRAN from UniVerse BASIC programs. This book is for experienced programmers who are familiar with UniVerse.

UniVerse ODBC Guide: Describes how to install and configure a UniVerse ODBC server on a UniVerse host system. It also describes how to use UniVerse ODBC Config and how to install, configure, and use UniVerse ODBC drivers on client systems. This book is for experienced UniVerse developers who are familiar with SQL and ODBC.

UV/Net II Guide: Describes UV/Net II, the UniVerse transparent database networking facility that lets users access UniVerse files on remote systems. This book is for experienced UniVerse administrators.

UniVerse Guide for Pick Users: Describes UniVerse for new UniVerse users familiar with Pick-based systems.

Moving to UniVerse from PI/open: Describes how to prepare the PI/open environment before converting PI/open applications to run under UniVerse. This book includes step-by-step procedures for converting INFO/BASIC programs, accounts, and files. This book is for experienced PI/open users and does not assume detailed knowledge of UniVerse.

API Documentation

The following books document application programming interfaces (APIs) used for developing client applications that connect to UniVerse and UniData servers.

Administrative Supplement for Uniclient APIs: Introduces IBM's seven common APIs, and provides important information that developers using any of the common APIs will need. It includes information about the UniRPC, the UCI Config Editor, the *ud_database* file, and device licensing.

UCI Developer's Guide: Describes how to use UCI (Uni Call Interface), an interface to UniVerse and UniData databases from C-based client programs. UCI uses ODBC-like function calls to execute SQL statements on local or remote UniVerse and UniData servers. This book is for experienced SQL programmers.

IBM JDBC Driver for UniData and UniVerse: Describes UniJDBC, an interface to UniData and UniVerse databases from JDBC applications. This book is for experienced programmers and application developers who are familiar with UniData and UniVerse, Java, JDBC, and who want to write JDBC applications that access these databases.

InterCall Developer's Guide: Describes how to use the InterCall API to access data on UniVerse and UniData systems from external programs. This book is for experienced programmers who are familiar with UniVerse or UniData.

UniObjects Developer's Guide: Describes UniObjects, an interface to UniVerse and UniData systems from Visual Basic. This book is for experienced programmers and application developers who are familiar with UniVerse or UniData, and with Visual Basic, and who want to write Visual Basic programs that access these databases.

UniObjects for Java Developer's Guide: Describes UniObjects for Java, an interface to UniVerse and UniData systems from Java. This book is for experienced programmers and application developers who are familiar with UniVerse or UniData, and with Java, and who want to write Java programs that access these databases.

UniObjects for .NET Developer's Guide: Describes UniObjects, an interface to UniVerse and UniData systems from .NET. This book is for experienced programmers and application developers who are familiar with UniVerse or UniData, and with .NET, and who want to write .NET programs that access these databases.

Using UniOLEDB: Describes how to use UniOLEDB, an interface to UniVerse and UniData systems for OLE DB consumers. This book is for experienced programmers and application developers who are familiar with UniVerse or UniData, and with OLE DB, and who want to write OLE DB programs that access these databases.

Chapter

Introduction

1

This chapter introduces transaction logging and recovery. The transaction logging system is a subsystem of UniVerse. It provides security against permanent data loss caused by events such as media or system failure. By restoring a backup copy of your files, then running roll-forward recovery (that is, applying logged updates from log files on disk or tape to your restored files), you can reinstate your files to a usable condition. Transaction logging preserves the application integrity of your UniVerse files by ensuring that any group of logically related updates to specified files is applied either in its entirety or not at all. Warmstart recovery restores the structural integrity of files after sudden system failure.

You use UniAdmin to administer the UniVerse transaction logging system on both UNIX and Windows servers. UniAdmin is fully documented in *Using UniAdmin*.

Transactions

To enable you to get your files back to a state of application-defined integrity, the transaction logging system uses *transactions*. UniVerse distinguishes two kinds of transaction:

- Data transactions
- Warmstart transactions

Data Transactions

A data transaction is a group of logically related file updates. The updates are intended to be processed as a single entity against the files. For example, a banking transaction that transfers funds from one account to another involves two logically related updates: a withdrawal and a deposit. To maintain a state of application integrity, both the deposit and the withdrawal must occur, or neither.

A set of related files moves from one state of transactional integrity to another by the complete processing of one or more transactions. Events that disrupt normal system operation can cause incomplete processing, resulting in loss of transactional integrity. You can always restore the transactional integrity of your files by using roll-forward recovery, because only completed transactions from the log file are applied to copies of your data files restored from backup.

When events terminate individual user processes in a controlled manner, transaction logging detects and reacts to the interruption of a transaction commit. Messages are sent and file operations are suspended pending restoration of transactional integrity.

Warmstart Transactions

Warmstart transactions log information about file-restructuring operations on specified files. When UniVerse is restarted after a system failure, these file-restructuring updates are applied to corrupted or damaged files, then all committed data transactions are applied, guaranteeing the integrity of the database.

Transaction Logging Tools

The transaction logging system provides you with several tools for its use. These include the following:

- UniVerse BASIC statements and functions
- UniAdmin menus and other administration tools
- UniVerse commands

UniVerse BASIC statements and functions let you define transactions in your application programs and determine the state of the transaction logging system. Chapter 2, "Transactions," discusses UniVerse BASIC transaction statements.

UniAdmin automates the administration, maintenance, and running of the transaction logging system.

UniVerse commands let you start and stop the transaction logging system, activate files for transaction logging, and create and manage log files. Chapter 8, "UniVerse Commands," discusses UniVerse commands used for transaction logging.

When to Use Transaction Logging

Transaction logging and recovery lets you recover data after certain kinds of media or system failure. Recovery can be relatively quick and automatic, or lengthy and manual, depending on the nature of the failure and how you are using transaction logging.

Media Failure

Media failure can render data stored on your system's storage media permanently inaccessible or unusable. A particle of dust, for example, can bring a read-write head into contact with the surface of a disk partition, destroying the whole disk. Unless your system is protected by duplicate hardware such as disk mirrors, you can use transaction logging to recover from such events after you repair or replace the damaged media.

You can reduce the impact of a media failure if you can restore a recent backup of UniVerse files to another disk. The impact on system availability is further minimized if you can easily apply updates made to the files after their last full backup to the files restored from the backup. Transaction logging lets you do this.

If you are logging file updates to disk, you can increase the probability of protecting current transactions if you store your UniVerse files on a different disk from the one where the log files are stored. Damage to a disk containing log files is not serious, provided the disk containing the UniVerse files remains intact. But you must back up your UniVerse files immediately, create a new set of log files on another disk, and restart transaction logging.

System Failure

Sudden system failure can result in updates to UniVerse files being incompletely recorded on disk. If the update involves file restructuring, the file may be structurally damaged and therefore unusable. Warmstart transaction logging and recovery automatically repairs any structural damage to specified files when the system is restarted after events such as power or system failures.

Scope of Transaction Logging

Data transaction logging provides a recovery capability for UniVerse files by the somewhat lengthy process of restoring file backups and rolling forward log files. You can use it in the event of media failure and of system failure. Warmstart transaction logging, however, provides a faster method of recovery from system failure.

Note*: Although log files produced by UniVerse transaction logging contain record update data in a form suitable for use in a displayable audit trail, we do not recommend or support such use of this data. Log files produced by subsequent revisions of the transaction logging system may not contain the roll-forward data in usable form.*

Transaction Logging and Performance

The recovery capability provided by transaction logging comes at some cost to system performance. Because transactions are stored in memory until committed or rolled back, memory demand is greater if you use transaction processing. Should the resulting increase in paging prove unacceptable, you may need more memory.

Transaction logging also needs additional CPU overhead to manage locks and log files. Transaction logging also increases the time spent writing to disk. Using UniVerse configurable parameters, you can fine-tune the amount of disk-writing overhead.

The numbers and data record sizes of updates in a transaction are dictated by the application. Large transactions cause the cache to move to disk, which can affect performance.

You can minimize performance costs by maintaining log files on disk drives where competing writes to disk are few, or on raw devices. Data transaction logging minimizes the performance impact of recoverability by writing only the content, not file-structure information, of updated records to the log files. (Warmstart transaction logging handles the logging of file-restructuring information.) Also, by saving your UniVerse files frequently, you can minimize the time it takes to restore them and run roll-forward recovery in the event the files are damaged.

Chapter

Transactions

2

This chapter discusses the following topics:

- What a transaction is
- UniVerse BASIC transaction statements
- File and record locking
- Restrictions while a transaction is active
- Transactions and sequential file I/O
- Transaction commit failures

What Is a Transaction?

A transaction is a group of logically related updates to UniVerse files. As mentioned in Chapter 1, "Introduction," a banking transaction that involves the transfer of funds from one account to another involves two logically related updates: a withdrawal and a deposit. Both updates, or neither, must be performed if the accounts concerned are to remain reconciled.

In UniVerse transaction processing, file updates are collected in a buffer as the application program requests them. Updates are applied to the files only when the program specifies that all the related updates have been requested (that is, requests transaction commit).

If transaction logging is active, file updates are written to the log file first, and then the updates stored in the buffer are issued against your UniVerse files. When any UniVerse BASIC statement following the transaction commit request is executed, it is certain that the log file writes have completed and the transaction can be processed by roll-forward recovery.

Because of further buffering done by the operating system, it may be several minutes before all the file updates are written to disk. This is why uncontrolled system failure can result in loss of transactional integrity in the data files. Warmstart recovery, when enabled, preserves database integrity in such cases, except for certain cases where file integrity is lost (see Chapter 3, "Transaction Logging.").

UniVerse BASIC Transaction Statements

In UniVerse BASIC programs transactions are delimited by transaction statements. The BEGIN TRANSACTION statement specifies the beginning of a transaction; the next COMMIT statement or ROLLBACK statement ends the transaction. The END TRANSACTION statement indicates where execution continues after the transaction is finished. (You can also use the PI/open versions of these statements: TRANSACTION START statement, TRANSACTION COMMIT statement, and TRANSACTION ABORT statement.) Program termination (by a STOP statement, ABORT statement, CHAIN statement, or END statement, not by a subroutine return) while a transaction is active executes a ROLLBACK statement.

Any update performed between the execution of a BEGIN TRANSACTION statement and a COMMIT statement is a *transactional update*. The record concerned must have been locked with an exclusive update record lock.

File and Record Locking

If you want to update a record in a transaction, first lock the record with an update record lock or an exclusive file lock. The READU statement, RECORDLOCKU statement, and FILELOCK statement impose these locks. To prevent inconsistencies arising from different users simultaneously updating the same record, you must lock the record with one of these locks before writing a new record to a UniVerse file as part of a transaction.

UniVerse also provides five transaction isolation levels to help you avoid certain data anomalies. When a transaction occurs, the transaction subsystem verifies that a program has acquired the required locks for the current isolation level. If it has not, the program fails. For information about Isolation Levels and Locks, see *UniVerse BASIC*.

The following example of *nontransactional* updating illustrates the kind of inconsistency that can result from the simultaneous updating of the same record by different users in the absence of an update record lock:

Sequence	**User One**	**User Two**
1	Read record A from disk into memory.	....
2	Update record A in memory.	....
3	....	Read record A from disk into memory.
4	Write record A from memory to disk.	....
5	....	Update record A in memory.
6	....	Write record A from memory to disk.

Nontransactional Updating Example

In this example, user one’s update to record A is invisible to user two, because user two’s read does not reflect the current state of the UniVerse file. User one’s update is lost.

Reads and Locks

As a general rule, if you intend to update a record transactionally, lock it with an update record lock while reading it. This ensures that any changes to the record affect the state of the record that your read indicates.

If you want to update a record without a read, you can use the UniVerse BASIC statement RECORDLOCKU. RECORDLOCKU sets an update record lock on a record without incurring the processing cost of a read.

Releasing Locks

All transactional updates to UniVerse files are deferred until the transaction is committed with the COMMIT statement or rolled back with the ROLLBACK statement. The release of a record lock that normally occurs with updates is likewise deferred, regardless of the type of update statement used, until the current transaction is committed or rolled back.

Note*: When a program executes the COMMIT statement, its deferred updates are applied to the appropriate files; then all locks are released. Alternatively, if a program executes the ROLLBACK statement, its deferred updates are discarded, then all locks obtained in the transaction are released. Attempts to use a RELEASE statement or FILEUNLOCK statement to release a lock being held for a transactional update, or to use a READL statement to demote such a lock, are ignored.*

Restrictions While a Transaction Is Active

The following restrictions apply while a transaction is active:

- For a given user, only one transaction can be active at a time.
- Executing any statement that releases a lock held on a transactionally updated record is ignored. For example:
  - A RELEASE statement with no file variable or record ID specified.
  - A RELEASE statement with a file variable specified but no record ID specified.
  - A RELEASE statement specifying a file variable and a record ID, if the record ID specifies a record that has already been transactionally updated.
  - A READL statement specifying a record that has already been transactionally updated (because the update record lock on the record would be demoted to a shared record lock).
  - A FILELOCK statement specifying a weaker file lock than one already acquired in a transaction.
  - A FILEUNLOCK statement (because maintenance of an update record lock on an updated record may depend on the file lock being maintained).
- You cannot update recoverable files on remote nodes through UV/Net.

Sequential File I/O

You can use the sequential file I/O statements READSEQ statement, WRITESEQ statement, and WRITESEQF statement without any restrictions during a transaction. They are not treated as part of the transaction; instead, they proceed as though the transaction were not active.

Transaction Commit Failures

Commit processing for a transaction can fail for the following reasons, among others:

- The disk fills up.
- A disk failure occurs.
- The user is forced out of UniVerse by the MASTER OFF command.
- Log files fill up, leaving active transactions unable to complete.

Events such as these can leave UniVerse files in an inconsistent state. The system detects this type of transaction commit failure and sends messages to the affected user terminal. In some cases, the transaction logging system automatically suspends processing of recoverable files.

Other events, such as power failures, can also result in inconsistent files because transactions are incomplete. These events cause system processing to stop in an obvious way. No explicit warning on possible inconsistencies is issued for such events. Warmstart transaction logging preserves file integrity in such cases (see Chapter 3, "Transaction Logging").

Chapter

3

Transaction Logging

This chapter covers the following topics:

- How transaction logging works
- Warmstart transactions and recovery
- Systemwide states of transaction logging
- States of log files
- The activity cycle of log files on disk
- The activity cycle of the log file on tape
- The UV_LOGS file
- The UV.TRANS file

How Transaction Logging Works

When transaction logging is active and enabled on your system, all updates to selected files are logged either to a special set of disk files (called *log files*, which are stored either in a log directory or on a raw device), or to a log file written directly to one or more tape devices. Transaction logging is implemented only for the files that you activate as *recoverable*.

If anything damages your UniVerse files, you can restore the most recent backup of your files from disk or tape and then run the transaction logging roll-forward utility. The roll-forward utility restores your files by reinstating everything successfully logged to disk or tape before the event that damaged them. Roll-forward does this by reapplying, in the original order, the necessary updates from the log file to your recoverable files. During a roll-forward, updates to files being recovered are prohibited.

Data Transactions and Warmstart Transactions

The system logs two kinds of changes made to recoverable files:

- Transactional and nontransactional data updates
- Warmstart transactions

Note*: When referring to logging updates to recoverable files, this book uses the term transaction to include both updates made within a transaction and updates made outside a transaction. Logging nontransactional updates is sometimes called data logging.*

Logging both transactional and nontransactional file updates preserves the integrity of your data. Warmstart transactions, on the other hand, log information about file-restructuring operations to preserve the structural integrity of your files. Warmstart transactions are described in "Warmstart Transaction Logging and Recovery" on page 11.

Logging File Updates

All logged file changes are stored in a log file. The log file can be stored:

- In a log directory on disk

- On a raw device
- On tape

If you log changes to a log directory or a raw device, all changes are logged to a set of one or more log files. If you log changes to tape, all changes are logged to one log file, which can be stored on one or more tape devices.

Logging to Disk or a Raw Device

When you log changes to a set of log files on disk or a raw device, only one log file is active at any given time. When the active log file fills up, the next log file in sequence becomes active and logging proceeds until all available log files fill up. When all log files are full, transaction logging is suspended and remains so until another log file is made available to record file changes. You can either create more log files as needed, if you have enough disk space, or you can back up all full log files to tape and then release them, making them available for reuse.

Logging to Tape

When you log changes to tape, all changes are logged to a single log file. This log file can be stored on one or more tape devices, only one of which is active at any given time. When the active tape device fills up, the next device in sequence becomes active and logging proceeds until all available devices fill up. When all devices are full, transaction logging is suspended and remains so until you change the tape media and release the tape devices, making them available for reuse.

Transaction Logging Modes

If you are logging changes to disk, you can set up the transaction logging system to log:

- Transactional and nontransactional file updates only
- Both file updates and warmstart transactions

To define which of these you want to log, you set two operating modes for transaction logging:

Mode	Used to log...
Archive	Transactional and nontransactional file updates
Checkpoint	Warmstart transactions, and transactional and nontransactional file updates

Transaction Loggin Operating Modes

You must run transaction logging in one of these modes, although both modes can be set to ON if you are logging file updates to disk.

***Note**: If you are logging changes to tape, you cannot use checkpoint mode. You can log only transactional and nontransactional file updates; you cannot log warmstart transactions*

Archive Mode

You use archive mode primarily for media recovery. In archive mode, updates to all recoverable files are logged, and transaction commits are logged either synchronously or at intervals specified by the UniVerse configurable parameters LOGSYCNT and LOGSYINT. Recovery involves restoring backup copies of recoverable files from disk or tape, then reapplying all logged updates in a roll-forward, using all log files from the time of the last backup to the point of failure.

Checkpoint Mode

You use checkpoint mode primarily for warmstart recovery. In checkpoint mode, the transaction logging system maintains a list of recoverable UniVerse files whose updates are logged in the current log file. When the log file is full, the checkpoint daemon issues an *fsync* command on each UniVerse file in the list, marks the log file as checkpointed, and releases it.

The advantage of checkpointing is that when warmstart recovery is needed:

- Recoverable UniVerse files need not be restored from the most recent backup.
- Log files need not be restored from the point of the most recent backup.

- Warmstart recovery has to roll forward only those log files that have not been checkpointed, rather than all log files made since the most recent backup.
- The system automatically detects the need for warmstart recovery and does it without the system administrator's intervention.

When recoverable UniVerse files are closed, the identifier (ID) of the last checkpoint is written to the file. (If checkpointing is not set to ON, no checkpoint ID is written.) This checkpoint ID, if present, can be used during media recovery to determine what is the oldest log file you need to recover the file.

When you back up files using the administration menus or the *uvbackup* command, the checkpoint ID is saved with the file. When you restore such files, they contain the checkpoint ID that was current when the files were backed up.

Stale Transactions

If transaction logging is running in checkpoint mode only, a particular kind of transaction can occur called a *stale transaction*. Stale transactions occur when log file space is limited and one or more very large transactions, or transactions that take a long time to complete, fill up all available log files before they are committed or rolled back. They can also occur if a user process is killed while a transaction is active. For information about stale transactions and how to deal with them, see Managing Stale Transactions in Chapter 6, "Managing Transaction Logging."

Flushing Log Buffer Contents to Disk

If you are logging to the log file on disk, by default the transaction logging system writes all log file updates synchronously from the log buffer to disk. You may not want your system to flush the log buffer to disk synchronously because this can slow down performance. On the other hand, flushing the log buffer at intervals rather than synchronously can result in the loss of logged updates if the system crashes before the current log buffer can be flushed to disk.

To configure how often log file updates are written to disk, you change the values of two UniVerse configurable parameters, LOGSYCNT and LOGSYINT. LOGSYCNT sets the number of transaction commits that should occur, and LOGSYINT sets the maximum time interval that should occur, before log buffer contents are written to disk.

Depending on the values of the LOGSYCNT and LOGSYINT parameters, the transaction logging system can be in three states:

- If LOGSYCNT and LOGSYINT are both set to 0, all log buffer writes to disk are synchronous.
- If LOGSYCNT and LOGSYINT are both set to values other than 0, the log buffer contents are synchronously written to disk when:
  - The number of commits specified by LOGSYCNT since the previous synchronous write is reached.
  - The number of seconds specified by LOGSYINT since the previous synchronous write is reached.
  - The current log file fills up.
  - The next warmstart transaction occurs, if checkpointing is set to ON.
- If LOGSYCNT is set to 0 and LOGSYINT is set to a value other than 0, the log buffer contents are synchronously written to disk when:
  - The number of seconds specified by LOGSYINT since the previous synchronous write is reached.
  - The current log file fills up.
  - The next warmstart transaction occurs, if checkpointing is set to ON.

Logging Directly to Log Files on a Raw Device

On UNIX systems, a raw device is a part of the disk that has no file system. On Windows platforms, a raw device is a disk with no file system. Logging updates to log files on a raw device can provide an improvement in performance because there is no file system overhead. For information about how to configure raw devices, see the documentation for your UNIX or Windows platform.

The Log Daemon

The log daemon, *uvlogd*, is an independent phantom process that logs UniVerse file updates to a log file.

Application processes put information to be logged in a log buffer. When a process is ready to commit a transaction, or when the log buffer is full, the log daemon flushes the log buffer to the currently active log file on disk or tape. When you are logging to disk, the flush can be synchronous, or you can set the LOGSYCNT and LOGSYINT configurable parameters to determine how often the log buffer should be flushed to disk.

You can specify a time frame in which to close the log file and move to the next available log file, even if the log file is not full. You define this time frame with the UVLOGSWITCH uvconfig parameter.

The UVLOGSWITCH parameter determines the time, in seconds, that the log file forces a switch to the next available log file if the current log file does not fill during the interval you specify. If you set this value to 0, UniVerse does not switch to the next log file until the current log file is full.

Note*: If the amount of time you specify expires and no logging activity has occurred (the log file is empty), UniVerse resets the time for the currently empty log file, ensuring that a completely empty log file is never marked as full.*

The log daemon handles error conditions. When an error occurs, a message is written to the logging information file in the log directory. If transactions can no longer be logged because the log files are full, a message is written to the logging information file and to the user's terminal, and all updates to recoverable files are frozen.

Restoring File Integrity

To restore file integrity you need two things:

- Tape backups of all recoverable files
- One or more log files containing all file updates to recoverable files made since the last full backup

If all the log files you need are on disk or a raw device, all you need to do is restore the most recent backup of your files, then roll forward the log files. If you have backed up some of the log files to tape, you must first restore all required log files to disk, then roll them forward. If you do not have enough room to restore all required log files, restore as many of them as you can (in sequence), roll them forward, then remove the restored log files. Restore the next group of log files and roll them forward, continuing until all file updates are rolled forward.

You decide which files you want to roll forward. You can roll forward the following:

- Particular files you specify
- Files specified in a select list
- All recoverable files

Protecting the Structural Integrity of Files

The UniVerse file system is also susceptible to problems when the structure of the file is changed. This vulnerability is most exposed in the following areas:

- File operations that attach and detach overflow and oversized blocks
- Split and merge operations on type 25 files
- Split and merge operations on dynamic files

In all three cases, two or more file system blocks must be written to disk before the file is consistent. Any system failure that interrupts this sequence of writes creates a UniVerse file inconsistency.

Warmstart Transaction Logging and Recovery

Warmstart transaction logging and recovery protects against database file corruption caused by a system failure. It does this by logging data about restructuring operations on selected files and stamping the files after the restructuring operations are successful. When UniVerse starts up after a system failure, the warmstart transaction in the log file and the stamp in the database file are used to check for any file corruption and to repair damaged files. In addition, data transactions in progress at the time of the failure are rolled back, and all committed transactions are guaranteed to be in the database.

Warmstart transaction logging is enabled when:

- The TXMODE configurable parameter is set to 1 (active).
- Checkpointing is set to ON.

For more information about setting the TXMODE parameter, see Activating the Transaction Logging System in Chapter 5, "Setting Up Transaction Logging."

Checkpointing

Checkpointing is a technique for limiting the number of logged updates that need to be rolled forward during a warmstart recovery. When a log file fills, files whose updates have been logged are flushed from the file system cache to disk. This guarantees that at warmstart you need to roll forward only the log files containing updates to files that have not yet been flushed to disk.

The checkpoint daemon builds a list of files that have been logged in the current log file. When that log file is full, or reaches the time frame you defined with the UVLOGSWITCH parameter, each of the logged files is flushed to disk, then the log file is marked as flushed to disk.

The log file is marked as checkpointed only when:

- All updated files listed in that log file have been flushed to disk.
- All transactions started before or during logging in that log file have been committed or rolled back.
- All updated files listed in other log files containing parts of such transactions have been flushed to disk.

When recoverable files are closed, the checkpoint ID of the last checkpoint is written to the header of the file. Roll-forward uses the checkpoint ID to determine the first log you need to recover the file. *uvbackup* and *uvrestore* also save and restore the checkpoint ID, so files backed up and restored using these utilities contain the checkpoint ID that was current when they were backed up.

The checkpoint ID in the file header can get “stale” if the file remains open or unused for a long time. For this reason you should use the checkpoint ID only as a guideline. You may have a better idea of the first log file you need for a roll-forward recovery.

System Requirements

Warmstart transaction logging and recovery is not supported on UNIX systems that do not have the *fsync* function.

Group buffer sizes and the alignment of UniVerse files must match the block sizes and the alignment of system files. Type 25 files require an 8K block size.

For warmstart, the group buffer size of recoverable hashed files (static and dynamic) must allow an integral number of groups to fit into a file block. For example, a file with a separation of 4 requires a file block size of 2K or a multiple thereof. Do not create files with a separation greater than the file block size.

Warmstart Transactions

Warmstart transactions are logged for all recoverable UniVerse files. They are created whenever a file update causes a file-restructuring operation to occur. Each file-restructuring operation is logged as a separate warmstart transaction. All warmstart transactions are logged synchronously to the log file when the UniVerse file structure is changed. Warmstart transactions are not affected by the LOGSYCNT and LOGSYINT configurable parameters.

Warmstart transactions differ from data transactions. File-restructuring operations are often compensating transactions within a primary transaction. They are formed and committed by the system, not by the user, and they change no file data. They are similar to nested transactions, except that warmstart transactions do not depend on whether the primary transaction is committed. If the primary transaction is rolled back, there is no need to roll back the warmstart transaction, since it simply ensures that the file structure is consistent.

Some warmstart transactions are primary transactions in cases where a file is activated for recovery but a nontransactional update triggers a restructuring of the file.

How Warmstart Recovery Works

Whenever UniVerse starts up, it creates a new shared memory segment and starts the log daemon. The log daemon determines if UniVerse shut down gracefully or terminated abnormally. If all log files are marked as checkpointed, it means UniVerse shut down gracefully and the UniVerse startup procedure continues. If UniVerse terminated abnormally, the log daemon identifies the oldest log file not marked as checkpointed and invokes warmstart recovery, which starts the roll-forward procedure.

Roll-forward occurs in three phases:

1. The roll-forward daemon checks all log files to be rolled forward and, for each UniVerse file, determines what is the last warmstart transaction.
2. Only warmstart transactions are processed in the second phase. The roll-forward daemon checks to see if all file-restructuring changes were completely flushed to disk. If needed, information logged in the warmstart transaction is used to repair any inconsistencies caused by an incomplete file-restructuring operation. When phase two is complete, all recoverable files are structurally consistent, even if data stored in the files is not.

3. Only nonwarmstart transactions are processed in the third phase. All logged updates and committed transactions are reapplied to the database in the order in which they originally occurred. When phase three is complete, all data in recoverable files is consistent with the state of the database after the last transaction committed before the system failed.

After all log files have been rolled forward, the UniVerse startup procedure continues.

Inconsistent Files

Sometimes a file cannot be rolled forward. When this happens, the file is marked as inconsistent, no further updates to that file are made, and warmstart recovery continues.

Files updates can fail for a number of reasons. When a file update fails, the *uvrolf.info* file (the roll-forward information file) is updated with the reason for the failure. To recover data in such cases, you should:

1. Resolve the problem with the file.
2. Clear the inconsistency flag.
3. Roll forward the file from disk or tape.

Warmstart Restrictions

Not all updates to recoverable files are currently logged. DEFINE.DF allocates blocks in a file's overflow region to hold partitioning information. RESIZE and CONFIGURE.FILE radically restructure files. CREATE.INDEX and DELETE.INDEX restructure logical files comprising multiple data files. The transaction logging system does not log these file-restructuring changes; therefore they cannot be recovered during a warmstart recovery. Before you use any of these commands on recoverable files, you must either suspend transaction logging or temporarily deactivate the file for logging.

Structural Integrity of Underlying System Files

The structural integrity of the underlying system files depends on various internal pointers and counters that organize the file's data. Structural integrity does not include the actual data stored in the file. A file's structural integrity must first be resolved at the operating system level.

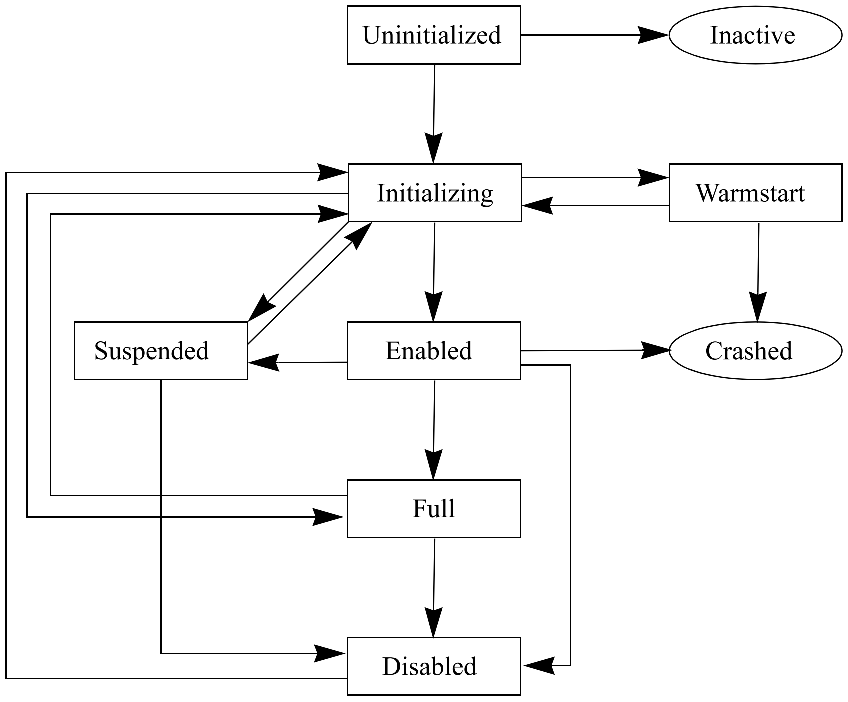
For example, suppose an overflow block is being added to a file when the system fails. A block must be retrieved from the free list and attached to the file. If the system crashes before the process is complete, the file system is inconsistent. Even if the file system is consistent, inconsistencies in the internal structure of a file can occur as a result of system failure during certain kinds of file restructuring, characterized by a requirement for multiple disk blocks to be written before the file is restored to a consistent state.

As a rule, UniVerse preallocates file space when the file is created. Since inconsistencies in the file system occur only when the structure of files is altered, preallocation minimizes the opportunity for such inconsistencies. It also helps to size your UniVerse files appropriately.

The operating system's reboot and recovery procedure runs *fsck* (on UNIX systems) or *chkdsk* (on Windows platforms) on all mounted file systems to verify and fix file inconsistencies. If *fsck* or *chkdsk* fixes a file, it may cause a problem at the UniVerse level. For example, blocks may be assigned that contain data to the *lost+found* directory (on UNIX systems) or to the *Found###* directory of the NTFS volume (on Windows platforms). This makes the file system consistent but leaves the file inconsistent at the application level. There is no general mechanism to verify the structural integrity of all UniVerse files. Since there is no centralized catalog of UniVerse files, warmstart checks only those files for which there are logged updates to be applied. Presumably files that were inactive at the time of the system failure are consistent. Of course, nonrecoverable files that were active may be broken.

Transaction Logging States

The following illustration shows all possible states that transaction logging can be in, and how each state can change to another state.



Transaction Logging States

The transaction logging system can be active or inactive. If it is active, the system can be in one of a number of states. Usually the log daemon manages the changes from one state to another. Certain error conditions (for example, when all available log files are full) can also change the system-wide state of transaction logging. The system administrator can also change the transaction logging state to enabled, disabled, or suspended (see "Logging Enabled, Disabled, or Suspended" on page 18).

Logging Uninitialized or Inactive

When UniVerse first starts up and creates a new shared memory segment, logging is *uninitialized*. The uninitialized state is transient and ends once the log daemon, *uvlogd*, starts up.

When the log daemon starts up, it reads the *uvconfig* file. If the TX MODE configurable parameter is set to 0 (inactive), the state of transaction logging changes from uninitialized to *inactive* and the log daemon exits.

If an application writes to a file when transaction logging is in the inactive state, the write succeeds, nothing is written to the log file (even if the application requests such a write), and no errors are issued. You can use the inactive state when you do not want to log updates. You can also use it to start UniVerse if a badly corrupted log file or data file prevents warmstart from completing successfully.

Transaction logging for recoverable files can proceed only after you activate transaction logging by setting the TXMODE configurable parameter to 1 (active).

Initializing and Warmstart States

When the log daemon starts up and finds the TXMODE parameter in the *uvconfig* file set to 1 (active), the log daemon changes the logging state from uninitialized or inactive to *initializing*.

When the log daemon is initializing, it checks to see if a warmstart recovery is needed. If it is, the log daemon changes the logging state from initializing to *warmstart*. When the warmstart completes, roll-forward sets the state back to initializing. For information about warmstart recovery, see "Warmstart Transaction Logging and Recovery" on page 11.

Logging Enabled, Disabled, or Suspended

Once transaction logging is active and initialized, it can be in the enabled, disabled (shut down), or suspended state. Transaction logging for recoverable files cannot proceed unless you enable transaction logging. The system administrator can change the transaction logging state to one of these states using the appropriate commands or menu options.

Logging Enabled

Normally, after the log daemon is initialized, it sets the logging state to *enabled*. The administrator can also enable transaction logging using the ENABLE.RECOVERY command or the transaction logging menus. In the enabled state, updates to recoverable files are written to your UniVerse files and to your log files.

Logging Disabled

The system administrator can disable transaction logging using the SHUTDOWN.RECOVERY command or the transaction logging menus.

If checkpointing is set to ON, the log daemon waits for the checkpoint daemon to finish checkpointing all log files before logging is fully disabled.

If transaction logging is running in archive mode, and you are logging file updates to tape, be sure you have adequate storage space on the currently active tape device before you shut down transaction logging. Adequate space ensures that the log buffer can be successfully flushed to tape before the log daemon shuts down. If the currently active tape fills up while the log buffer is being flushed, and if there are no more tape devices available, the transaction logging system enters the full state. For information about what to do when this happens, see Releasing Full Tape Devices in Chapter 6, "Managing Transaction Logging."

In the disabled state, updates to recoverable UniVerse files are written to your UniVerse files but not to your log files.

Warning*: You cannot recover updates made while transaction logging is disabled. To fully reinstate transaction logging, you must stop system operations, back up your files, and reenable transaction logging. See Restarting Transaction Logging in Chapter 6, "Managing Transaction Logging," for how to do this.*

Logging Suspended

The system administrator can suspend transaction logging using the SUSPEND.RECOVERY command or the transaction logging menus. When transaction logging is suspended, updates to recoverable files are disallowed for the duration of the suspension.

Logging is automatically suspended when all log files are full.

When transaction logging is suspended, and if all log files have been fully checkpointed, you can roll forward updates to UniVerse files using the transaction logging menus. The integrity of your UniVerse files is maintained by prohibiting updates to recoverable files while transaction logging is suspended.

Logging Full

The transaction logging state changes to full when there are no more available log files for logging file updates. When transaction logging is in the full state, you must make more log files available before you can log more file updates.

Logging to Disk or a Raw Device

The log daemon sets logging to the full state when all log files are full and there are no more empty log files, or when certain kinds of stale transaction occur. In the full state, the log daemon sends a message to the logging information file, *uvlogd.info*, and exits. User processes that write to the log file while it is in the full state will pause until the state changes.

To start transaction logging again, you must back up all full log files to tape, release them for reuse, then enable the transaction logging system.

Logging to Tape

The log daemon sets logging to the full state when no available tape device is found in the active device list. This can happen if all defined tape devices fill up and are not released (made available again). User processes that write to the log file while it is in the full state will pause until the state changes.

To start transaction logging again, you must remove all full tape media, reload the tape devices, release them for reuse, then enable transaction logging.

Logging Crashed

The logging state changes to crashed when an unrecoverable error occurs or when roll-forward encounters a serious problem during warmstart. The message displayed when this occurs is “Suspended—Internal Error.” When logging is in the crashed state, you cannot restart the log daemon from the transaction logging menus. You must shut down UniVerse completely and restart it, creating a new shared memory segment.

If the underlying disk system used for logging is damaged, restarting UniVerse may put the logging system into the crashed state again. In this case you must set the TX MODE configurable parameter to 0 (inactive) in the *uvconfig* file before you can restart UniVerse. This solution compromises the integrity of your system, but you may find it useful in emergencies. After you correct the problems blocking the warmstart, reset TXMODE to 1 (active).

How File Updates Are Treated

The following table shows how file updates are treated in each transaction logging state. Transactional updates are those included in a transaction. Recoverable files are those activated for transaction logging.

Logging State	Update not in transaction, file is not recoverable	Update not in transaction, file is recoverable	Transactional update, file is not recoverable	Transactional update, file is recoverable
Inactive	Update succeeds, nothing is logged.	Update succeeds, nothing is logged.	Update succeeds, nothing is logged.	Update succeeds, nothing is logged.
Enabled	Update succeeds, nothing is logged.	Update succeeds and is logged.	Update succeeds, nothing is logged, and a warning is displayed.	Update succeeds and is logged.
Suspended Full Initializing Warmstart	Update succeeds, nothing is logged.	Update waits until logging is enabled.	Update waits until logging is enabled.	Update waits until logging is enabled.
Disabled	Update succeeds, nothing is logged.	Update succeeds, nothing is logged.	Update fails.	Update fails.
Crashed	Update succeeds, nothing is logged.	Update fails.	Update fails.	Update fails.

Effect of Logging State on File Updates

Log File States

Do not confuse the system-wide state of transaction logging with the state of a particular log file. Each log file has its own state and at any time can be Available, Current, NeedsSync, Full, or Released.

If your system is running with NLS enabled, each log file is marked as an NLS log file. Log files generated on systems with NLS turned on cannot be rolled forward with NLS turned off. Log files generated on systems with NLS turned off cannot be rolled forward on systems with NLS turned on.

Available

A log file's status is Available when it is first created and contains no updates. Even when NLS is turned on, Available log files are not marked as NLS log files until they are used (that is, until they become Current).

Current

A log file's status is Current if it is the file to which updates are currently being written. At any time, only one log file is Current. If you are logging file updates to multiple tape drives, do not confuse the current tape drive with the current log file. There is no correspondence between log files and tape devices: there is only one log file on tape, and it is Current. However, only one tape device can be active (or current) at any given time.

NeedsSync

This status is used only if you are logging file updates to disk and checkpoint mode is set to ON. A log file's status is set to NeedsSync when it has no more room available for writing updates. After the checkpoint daemon has fully checkpointed the log file, its status changes to Full if archive mode is set to ON. If archive mode is set to OFF, the status normally changes to Released. When certain stale transactions occur, however, the state can change to Full. For information about handling stale transactions, see Managing Stale Transactions in Chapter 6, "Managing Transaction Logging."

Full

Normally, a log file’s status is Full when it has no more room available for the writing of updates. If you are logging file updates to disk, as soon as the Current log file becomes Full, transaction logging switches to the next Available log file.

If you are logging file updates to tape, the tape log file becomes Full only when all tape devices are full and there are no more available tape devices. When this happens, the transaction logging system also enters the full state (see “Logging Full” on page 20).

If archive mode is set to OFF, the Current log file’s status does not change to Full. The only exceptions are:

- When something goes wrong during a warmstart recovery and the system wants to preserve the current log file for a roll-forward
- When certain kinds of stale transaction occur and user intervention is needed

Released

This status is used only if you are logging file updates to disk or a raw device. A log file’s status is Released when the updates it contains are no longer required. If archive mode is set to OFF, and if the archive bit in the log file is not set, the checkpoint daemon releases the log file for reuse once it is fully checkpointed.

You can also release a log file with the RELEASE.LFILE command once the updates it contains are no longer required—that is, when the contents of the log file have been backed up to disk or tape.

A Released log file no longer exists physically on disk.

Activity Cycle of Disk Log Files

Log files are sequentially numbered and named in the form *lg1*, *lg2*, *lg3*, and so on. A log file begins its activity cycle as Available; it then becomes Current, then Full (or NeedsSync, then Full), then Released.

To avoid confusion, no log file sequence number is used more than once. When a log file is created or released, it is renamed by assigning the next available unused sequence number. Transaction logging menus and scripts that facilitate backup of Full log files automatically rename the log files.

Example

This example illustrates the activity cycle of five log files. Immediately after creation, all log files are Available. When transaction logging starts, the log files are in the following states:

lg1	Current
lg2	Available
lg3	Available
lg4	Available
lg5	Available

When Archive Mode Is ON and Checkpoint Mode Is OFF

When the first log file becomes Full, transaction logging switches automatically to the next Available log file and logging continues:

lg1	Full
lg2	Current
lg3	Available
lg4	Available
lg5	Available

When the second log file becomes Full, transaction logging switches automatically to the next Available log file and logging continues:

lg1	Full
lg2	Full
lg3	Current
lg4	Available
lg5	Available

When the third log file becomes Full, transaction logging switches automatically to the next Available log file and logging continues:

lg1	Full
lg2	Full
lg3	Full
lg4	Current
lg5	Available

Given this example, Full log files might be backed up to tape. Full log files are backed up to tape and released, as follows:

lg1	Released
lg2	Released
lg3	Released
lg4	Current
lg5	Available
lg6	Available (formerly lg1)
lg7	Available (formerly lg2)
lg8	Available (formerly lg3)

The Released log files are Available again and renumbered so their sequence numbers begin at one higher than the highest-numbered log file currently Available.

When Archive Mode Is OFF and Checkpoint Mode Is ON

When the first log file goes into the NeedsSync state, transaction logging switches automatically to the next Available log file:

lg1	NeedsSync
lg2	Current
lg3	Available
lg4	Available
lg5	Available

When the NeedsSync log file is flushed to disk, the log file is Released for reuse and a new Available log file is created:

lg1	Released
lg2	Current
lg3	Available
lg4	Available
lg5	Available
lg6	Available (formerly lg1)

As the examples illustrate, multiple log files let you maintain extensive storage for logged updates, with only parts of it residing continuously on the system. A related advantage is that you can back up one or more log files without interfering with the operation of the transaction system.

Activity Cycle of the Tape Log File

File updates are logged to one log file on tape regardless of how many tape devices you are using. When a tape device becomes the current archiving device, it is moved from the front of the defined device list to the end of the list. This lets you cycle through the defined tape devices as long as they are available or free. Once a tape fills up, its tape device is marked as full, and it cannot be used until it is released. If all defined tape devices fill up and are not released, the transaction logging system goes into the full state. For a description of the procedure for releasing tape devices and making them available again, see Releasing Full Tape Devices in Chapter 6, "Managing Transaction Logging."

The UV_LOGS File

The UV_LOGS file is in the UV account. It maintains information about the state of the log files in the log directory. The UV_LOGS file is referenced by the transaction logging menus.

The principal task of the UV_LOGS file is to maintain information about log files that have been backed up to tape and may be required for application to the UniVerse files during roll-forward recovery. UV_LOGS contains a record for each log file that has been created or released. The following table shows the format of a record in the UV_LOGS file. UniAdmin displays the contents of the UV_LOGS file in the Transaction Logging window.

Field	Name	Description
0	@ID	Record ID. The sequence number of the log file. The synonym SEQUENCE.NUMBER refers to this field.
1	START.DATE	The date that the log file became Current. START.DATE is used by the roll-forward utility to determine the relevance of log files that have been released and (presumably) backed up.
2	START.TIME	The time that the log file became Current.
3	FULL.DATE	The date that the log file became Full. The log daemon, or (if it is set to ON) the checkpoint daemon, copies the date the log file became Full to the FULL.DATE field.
4	FULL.TIME	The time that the log file became Full. The log daemon, or (if it is set to ON) the checkpoint daemon, copies the time the log file became Full to the FULL.TIME field.
5	SIZE	The size of the log file in kilobytes. This field is intended for customer use. It provides a means of determining how much disk space is needed to accommodate a restored log file.
6	*None*	Current status of the log file.

UV_LOGS Record Format

Field	Name	Description
7–19	*Reserved*	These fields are reserved for future enhancements and are not to be used.
20	USER.DATA	This field is available for user data.
21+	*Reserved*	All fields from this point on are reserved for future enhancements and are not to be used.
—	STATUS	This field, defined by an I-descriptor, contains the current condition of the log file.

UV_LOGS Record Format (Continued)

The STATUS field of the UV_LOGS file can have the following values:

Value	Description
Released	Released log files no longer exist on disk.
Full	The log file is full.
Available	The log file is available for use.
Current	The log file is currently active.
Error	The log file has an error.
NeedsSync	Not yet processed by checkpointing.

UV_LOGS STATUS Values

The CREATE.LFILE command writes a new record to UV_LOGS for each new log file it creates. The RELEASE.LFILE command then updates this record for the log files it releases and subsequently creates a new record for the log files it creates, switching STATUS fields as required. The DELETE.LFILE command deletes empty (that is, Available) log files from the log directory and removes the corresponding record from UV_LOGS.

The UV_LOGS file is referenced by the following:

- The roll-forward utility, to determine from which log file to take updates when performing recovery
- The UniAdmin Transaction Logging window
- The log daemon, to synchronize log entries after a logging system failure

The following commands and menu options do not change the UV_LOGS file:

- LOG.RESTORE command
- DEL.RFILE command
- The UniVerse Admin Restore…, Delete…, and Rollfwd options (Recover Files window)
- The UniAdmin Rollfwd option (Recover Files From Tape window)

The file dictionary of UV_LOGS includes the following user records:

- CHECKPOINT, which indicates the current status of checkpoint mode
- ARCHIVE, which indicates the current status of archive mode and type
- LOGS.DIR, which contains the pathname of the current log directory
- LOG.NEXT, which contains the number of the next log file to be created

The UV.TRANS File

The UV.TRANS file is in the UV account. It maps identification numbers of recoverable files to the paths of the files. Recoverable file identification numbers are used in log files to associate updates with the relevant recoverable files. A record exists in the UV.TRANS file for every recoverable file. A mapping file eliminates the need to put in log files the full paths of the recoverable files to which updates relate. This is desirable because:

- Recoverable file identification numbers occupy less space than the full paths of recoverable files. This minimizes the amount of data written to log files and therefore optimizes performance.
- If the name or path of a recoverable file changes, it is simple to change the mapping information in the UV.TRANS file to reflect the new name or path. Updates to the recoverable file can still be applied. Users should therefore be aware of the impact on the UV.TRANS file whenever they copy, move, back up, restore, delete, and recreate recoverable files.

UniVerse automatically updates the UV.TRANS file:

- When you activate a file for transaction logging. A record is added to the UV.TRANS file for the newly activated file.
- When you deactivate a file for transaction logging. Field 3 in the corresponding record of the UV.TRANS file is set to 0 to prevent logging of updates.

UniVerse reads the UV.TRANS file:

- When you do a roll-forward. The UV.TRANS file is checked to establish the paths of recoverable files that correspond to the recoverable file identification numbers written to updates in the log files.
- When recoverable files are opened to check that the path of the file being opened matches the path recorded for that recoverable file identification number in UV.TRANS.

The following table shows the format of a record in the UV.TRANS file.

Field	Name	Description
0	@ID	Record ID. The identification number of the recoverable file. This number is kept in the X-descriptor CURR.COUNT in the dictionary of the UV.TRANS file. It is also kept in the header of the recoverable file.
1	ACCOUNT	The name of the account.
2	FILE	The name of the file.
3	*None*	Activation status.
4	PATH	The full path of the recoverable file.
—	STATUS	This field, defined by an I-descriptor, returns an empty string if field 3 is 0; otherwise it returns Active.

UV.TRANS Record Format

When, through the progressive recycling of log files, the updates pertaining to a deleted recoverable file no longer exist, the record in UV.TRANS corresponding to that deleted recoverable file can be deleted. When you use UniAdmin, this happens automatically.

The UV.TRANS file can also be used as a convenient means of locating recoverable files on the system.

Managing the UV.TRANS File

UV.TRANS is a system file used internally to direct the processes of logging and recovery. The behavior of roll-forward is determined by the content of UV.TRANS at the time the roll-forward begins processing. Because the roll-forward utility consults the UV.TRANS file for mapping information about the recoverable files to which updates will be applied, it is imperative that the UV.TRANS file reflect the current state of the recoverable files on the system.

***Note**: Be sure you always have a backup copy of the current UV.TRANS file. Whenever UV.TRANS changes (for example, when you activate or deactivate a file for transaction logging), you should back up the UV.TRANS file.*

Chapter

The Transaction Logging Window

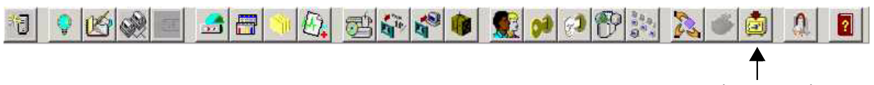
4

Use the **UniVerse Transaction Logging** dialog box to set up and manage transaction logging.

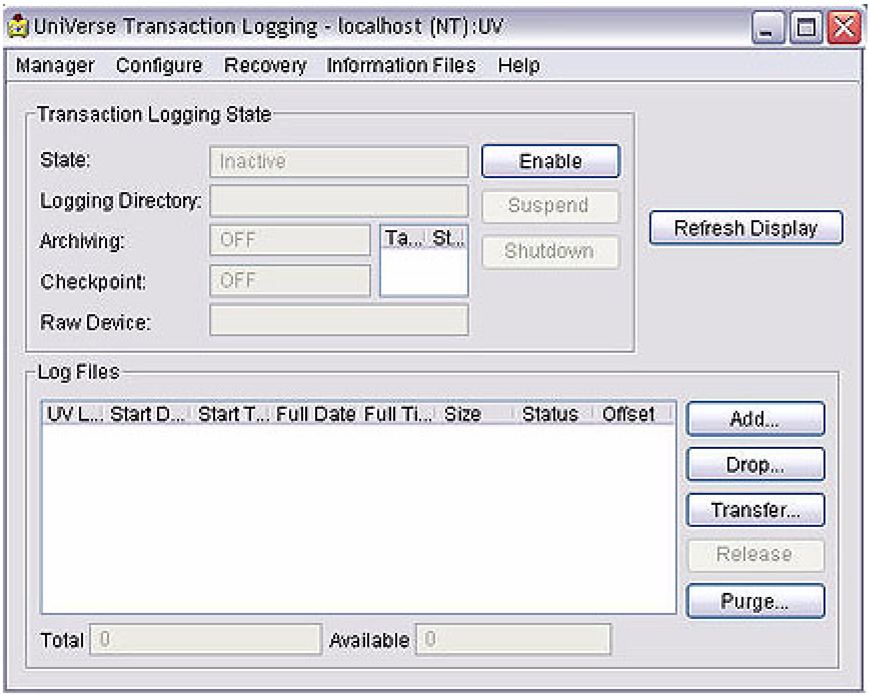
Note*: For detailed information about setting up and managing UniVerse Transaction Logging through UniAdmin, see Using UniAdmin.*

Select one of the following methods to access the **UniVerse Transaction Logging** dialog box:

- From the **UniAdmin** window, double-click **Transaction Logging**.
- From the **UniAdmin** menu, select ***Admin,*** then click **Transaction Logging**.
- From the **UniAdmin** toolbar, click the **Manage Transaction Logging** icon, as shown in the following example.



The following example illustrates the **UniVerse Transaction Logging** dialog box:


UniVerse Transaction Logging - localhost (NT):UV
Manager
Configure
Recovery
Information Files
Help
Transaction Logging State
Enable
Suspend
Shutdown
Refresh Display
Ta... St...
Log Files
UV L... Start D... Start T... Full Date Full Ti... Size Status Offset
Add...
Drop...
Transfer...
Release
Purge...

State: Inactive
Logging Directory:
Archiving: OFF
Checkpoint: OFF
Raw Device:
Total 0
Available 0

***Note**: The Transaction Logging window displays the state of transaction logging when the task was activated. To display the current logging state, click **Refresh Display**.*

The tasks you can perform from this window include:

- Configuring transaction logging
- Activating recoverable files
- Managing transaction logging
- Recovering files
- Viewing and deleting information files

Chapter 5, "Setting Up Transaction Logging," and Chapter 6, "Managing Transaction Logging," explain how to perform these tasks.

The Transaction Logging Window

The Transaction Logging window has the following components:

- Menu bar. There are five pull-down menus:
  - **Manage**. There are nine options on this menu. If you are logging to disk, all options except **Release Tape** are available. If you are logging to tape, the **Add Log Files**, **Drop Log Files**, **Transfer Log Files**, **Purge Log Files**, and **Release Log File** options are not available.

    ***Note**: You must choose a log file to release before you can use the **Release Log File** option.*
  - **Configure**. You set up transaction logging, and specify which UniVerse files are recoverable using the **Logging** and **Recoverable Files** options.
  - **Recovery**. You set up recovery from disk or from tape using **Rollforward from Disk** and **Rollforward from Tape**. You can also clear the inconsistency flag by choosing **Clear File Inconsistency Flag**.
  - **Information Files**. You view or delete the information files using the six options on this menu.
  - **Help**. Choose **Contents** to display the UniAdmin Help Contents, or choose **Transaction Logging** to display the Transaction Logging Help Topic.
- Buttons. There are nine buttons, eight of which perform the same tasks as the Manage menu options. The **Refresh Display** button updates the Transaction Logging State and the Log Files areas.

- Transaction Logging State area. It contains the following fields, which cannot be edited:
  - **State.** The current logging status. There are many different logging states. The system administrator can change the state to enabled, disabled, or suspended.
  - **Logging Directory.** The path of the directory containing the log files. See Configuring the Transaction Logging System in Chapter 5, "Setting Up Transaction Logging," for information about how to change the logging directory.
  - **Archiving.** The state of archiving: Off, To Disk, or To Tape. If you are logging to tape, two additional columns appear. **Tape Device** specifies the devices when transaction logging was configured. **Status** shows the current state of each tape device. If you are logging to tape, the status is In Use. If the tape is full, the state is Full.
  - **Checkpoint.** The state of checkpoint: ON or OFF.
  - **Raw Device.** The path of the raw device containing the log files. See Setting Up or Changing the Transaction Logging Configuration in Chapter 5, "Setting Up Transaction Logging," for information about how to change the raw device path.

- Log Files area. It displays the state of the log files. The information for each entry is read from the UV_LOGS file, and includes:
  - **UV Log.** The log file sequence number, which is unique to each log file that is created.
  - **Start Date.** The date that transaction logging started logging to this file.
  - **Start Time.** The time that transaction logging started logging to this file.
  - **Full Date.** The date when the log file became full.
  - **Full Time.** The time at which the log file became full.
  - **Size.** The size (in bytes) of the log file.
  - **Status.** The current status of the log file:

    Available. The log file contains no updates and is available for logging.

    Current. The log file is being used for logging and is being written to with file updates.

    Full. The log file has no room left for the writing of updates. When logging to disk, the next available log file is then automatically used to continue logging.

    NeedsSync. The log file has no room left for the writing of updates, and the Checkpoint option is set to ON. The next status in the sequence depends on whether the Archive option is checked. If Archive is set to ON, the status updates to Full. If Archive is set to OFF, the status updates to Released.

    Released. This status occurs only if you are logging to disk, and is set when the log file is released or transferred, that is, when the updates it contains are no longer required.
  - **Offset.** The starting offset where log files on the raw device begin.

The following summary information is also displayed:

- **Total.** The total number of bytes required for the log files.
- **Available.** The total number of bytes available on the system.

Chapter

5

Setting Up Transaction Logging

This chapter describes how to set up the transaction logging system. There are two general setup procedures, one if you are logging updates to disk, the other if you are logging updates to tape.

You set up transaction logging using the Transaction Logging window of UniAdmin (see Chapter 4, "The Transaction Logging Window").

The next section describes how to set up transaction logging to disk. For information about setting up transaction logging to tape, see "Setting Up Transaction Logging to Tape" on page 11.

Setting Up Transaction Logging to Disk or Raw Device

To set up transaction logging to disk or raw device, do the following:

1. Create a log directory.
2. (Optional) Specify the path of a raw device for storing log files.
3. Set the transaction logging modes.
4. Create one or more log files in the log directory or on the raw device.
5. (Optional) Configure how frequently to write the contents of the log buffer to disk.
6. Activate the transaction logging system (see "Activating the Transaction Logging System" on page 14).
7. Activate UniVerse files for transaction logging, making them recoverable (see "Activating UniVerse Files for Recovery" on page 16).
8. Enable the transaction logging system (see "Enabling the Transaction Logging System" on page 22).

The following sections describe in detail how to do steps 1 through 5.

Configuring the Transaction Logging System

Before you enable and use the transaction logging system, you must configure it. This can include:

- Creating a log directory where log files are to be stored
- Specifying a raw device where log files are to be stored
- Setting the transaction logging mode

***Note**: You also need to specify which UniVerse files are recoverable and create log files before you can enable logging.*

Creating the Log Directory

UniVerse can write recoverable file updates to log files stored either in a log directory or on a raw device (on UNIX systems also known as a character special device).

Before updates to recoverable files can be written to log files on disk, you must first create a log directory. The log directory will contain three transaction logging information files in addition to any log files you create. If you plan to keep your log files on a raw device, you must also specify a path for the raw device.

For best performance and data protection, put the log directory on a different disk drive from the one holding your UniVerse files. This precaution minimizes the effect of a media failure since damage to a single disk drive may result in either the loss of your UniVerse files or the loss of your log files, but not both.

Setting Transaction Logging Modes

You can set two operating modes for transaction logging to disk:

- Archive mode
- Checkpoint mode

To log...	Set checkpoint mode to...	Set archive mode to...	Set archive type to...
Only transactional and nontransactional file updates	OFF	ON	DISK
Both file updates and warmstart transactions	ON	ON or OFF	DISK

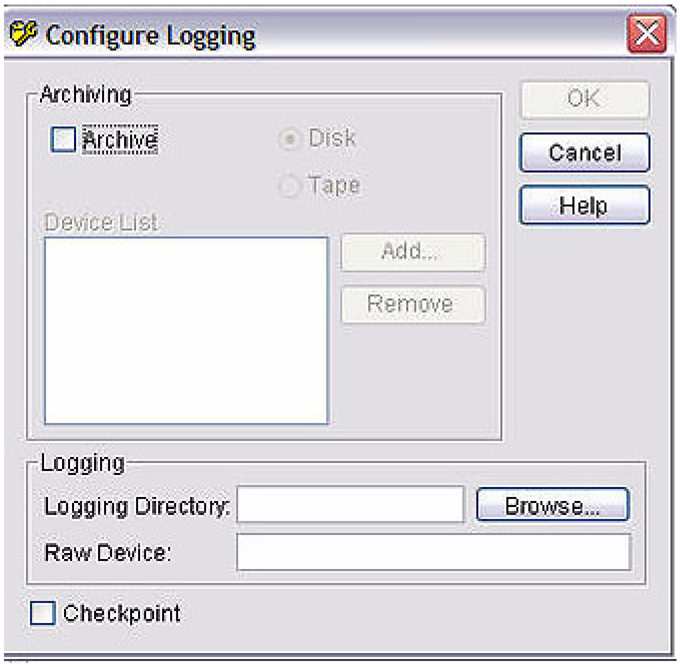
Transaction Logging Modes

For information about archive and checkpoint modes, see Transaction Logging Modes in Chapter 3, "Transaction Logging."

Setting Up or Changing the Transaction Logging Configuration

To set up or change the configuration for transaction logging to disk:

1. Choose **Configure** -> **Logging**. The **Configure Logging** dialog box appears, as shown in the following example:



2. (Optional) Select the **Archive** check box if you want to perform archiving.
3. Click the Disk option button.
4. (Optional) Select the **Checkpoint** check box. You can choose this type of transaction logging in addition to, or instead of, the **Archive** check box.
5. Enter the path of the logging directory in the **Logging Directory** field. You can also use **Browse** to search the system for a suitable directory. If the directory whose path you enter does not exist, it is created. If the directory already exists, a message box confirms that this is the directory to use. Click **Yes** to use the directory.
6. (Optional) Enter the path of the raw device where you want to store log files. On Windows platforms, the syntax is \\.*physicaldrive*n. On UNIX the syntax varies depending on which platform you are using. See your UNIX documentation for more information.
7. Click **OK** to save the settings and to close the dialog box.

***Note**: To reconfigure transaction logging on a system where it is already enabled, you must shut down transaction logging before you choose the **Logging** option.*

Adding and Dropping Log Files

How many log files you need is determined partly by the transaction logging modes you set (see "Setting Transaction Logging Modes" on page 5). It also depends on whether you use a raw device to store your log files.

Normally you need more than one log file. If the current log file fills up and the transaction logging system has no other log file to switch to, transaction logging is automatically suspended system-wide and updating recoverable files is prohibited. While this is not a threat to file integrity, it can be inconvenient.

Although in some cases it is possible to have a single log file, we do not recommend it. With more than one log file, you can back up all full log files (except the currently active log file) at any time. Transaction logging remains enabled because it can write to another log file.

The size of the log files you create depends on the volume of updates you expect. See Calculating the Volume of Logged Data in Chapter 6, "Managing Transaction Logging," for how to calculate this volume. Be generous in determining the size of your log files, because if you run out of log file space, the transaction logging system stops.

***Note**: If you store your log files on a raw device, the total amount of disk space occupied by the log files must not exceed 2 gigabytes.*

The log files you create have names like *lg1*, *lg2*, and so on. Do not put files other than log files in the log directory, because the performance of transaction logging may be impeded. Log files are preallocated on disk at the time they are created, so be sure you have enough disk space before you create them.

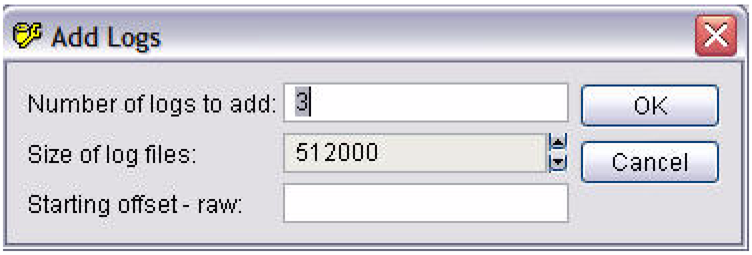
When transaction logging is enabled, updates are written to the first log file, *lg1*. When this file is full, transaction logging switches to the next log file, *lg2*. When this file is full, logging switches to the next log file, and so on.

Adding Log Files

Complete the following steps to create log files:

1. Do one of the following:
   - Click the **Add…** button.
   - Choose **Manager -> Add Log Files…** .

   The Add logs dialog box appears:



Number of logs to add: 3
Size of log files: 512000
Starting offset - raw:

2. Enter the number of log files to add or create in the **Number of logs to add** field. The default is 3.
3. Do one of the following:
   - If you are logging to the log directory, enter the number of bytes for each log file in the **Size of log files** field. This number is always rounded to the nearest increment of 512 bytes. You can use the arrows to increase or decrease this value. The default is 512000.
   - If you are logging to a raw device, enter the starting offset in the Starting Offset - Raw field. This number must be a multiple of 512, or 0. If you are adding more log files to a raw device that already contains log files, the next starting offset is automatically filled in.
4. Click **OK**. The log files are created (with a status of Available) and are added to the Log Files list in the Transaction Logging window.

Dropping Log Files

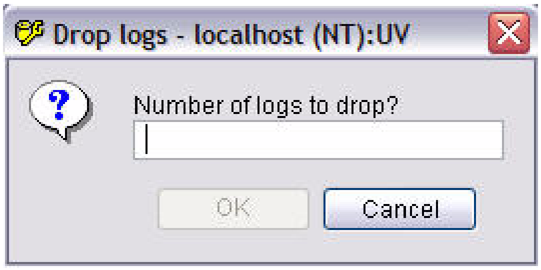
If you are logging to the log directory, you can delete (drop) unneeded log files if the status is inactive and the log files have not been used. Once log files have been dropped, the log file sequence numbers become available and are used by the next log files you create.

If you store your log files on a raw device, you can drop unneeded log files with a status of Available when the transaction logging system is suspended or shut down. While transaction logging is enabled, you can drop only Available log files whose offset is higher than the current log file.

To drop a log file:

1. Do one of the following:
   - Click the **Drop** button.
   - Choose **Manager -> Drop Log Files**.

   The Drop logs dialog box appears.



2. Enter the number of log files to drop in the **Number of logs to drop** field.
3. Click **OK**. The log files with the highest log file specification numbers are deleted. If you specify a number of log files greater than the number available, only those available are deleted.

Configuring the Frequency of Log Buffer Writes to Disk

You can configure how often log file updates are written to disk by changing the values of three configurable parameters in the *uvconfig* file, LOGSYCNT and LOGSYINT, and UVLOGSWITCH

Setting the Maximum Number of Transaction Commits

The LOGSYCNT parameter specifies the maximum number of commits that should occur between each flush of the log buffer. The default value for this parameter is 0, which means all updates are written synchronously. It does not make sense to specify a value of 1, because each commit generates a file system synchronization call, which is more expensive than a synchronous write. So you should set the value of this parameter to 2 or more.

The LOGSYCNT parameter has no effect on a nontransactional database environment.

Setting the Maximum Time Interval Between Log Buffer Writes

The LOGSYINT parameter specifies the maximum time interval in seconds between each flush of the log buffer. The default value for this parameter is 0 (updates are written synchronously), unless the value of LOGSYCNT is nonzero: in that case, LOGSYINT defaults to 120 seconds. A nonzero value less than 5 is automatically set to 5.

***Note**: In some cases if the values of LOGSYCNT and LOGSYINT are set too low, performance will be slower than if the values are set to 0.*

Specifying a Time Frame to Close the Log File

You can specify a time frame in which to close the log file and move to the next available log file, even if the log file is not full. You define this time frame with the UVLOGSWITCH uvconfig parameter.

The UVLOGSWITCH parameter determines the time, in seconds, that the log file forces a switch to the next available log file if the current log file does not fill during the interval you specify. If you set this value to 0, UniVerse does not switch to the next log file until the current log file is full.

***Note**: If the amount of time you specify expires and no logging activity has occurred (the log file is empty), UniVerse resets the timer for the currently empty log file, ensuring that a completely empty log file is never marked as full.*

Setting Up Transaction Logging to Tape

To set up transaction logging to tape on your system, do the following:

1. Create a log directory.
2. Set the archive transaction logging mode to ON.
3. Set the archive type to Tape.
4. Specify the names of the tape devices to log to.
5. Activate the transaction logging system (see "Activating the Transaction Logging System" on page 14).
6. Activate files for transaction logging, making them recoverable (see "Activating UniVerse Files for Recovery" on page 16).
7. Enable the transaction logging system (see "Enabling the Transaction Logging System" on page 22).

The following sections describe in detail how to do steps 1 through 4.

Creating the Log Directory

Before updates to recoverable files can be written to log files on tape, you must first create a log directory on disk.

Setting the Transaction Logging Mode

If you are logging file updates to tape, you can set only the archive mode to ON. Checkpoint mode is automatically disabled if you set the archive type to Tape.

You must also specify the names of the tape devices to log to. Updates are logged in 512-byte blocks to multiple tape devices in the order specified. Updates are cached until a 512-byte block is filled.

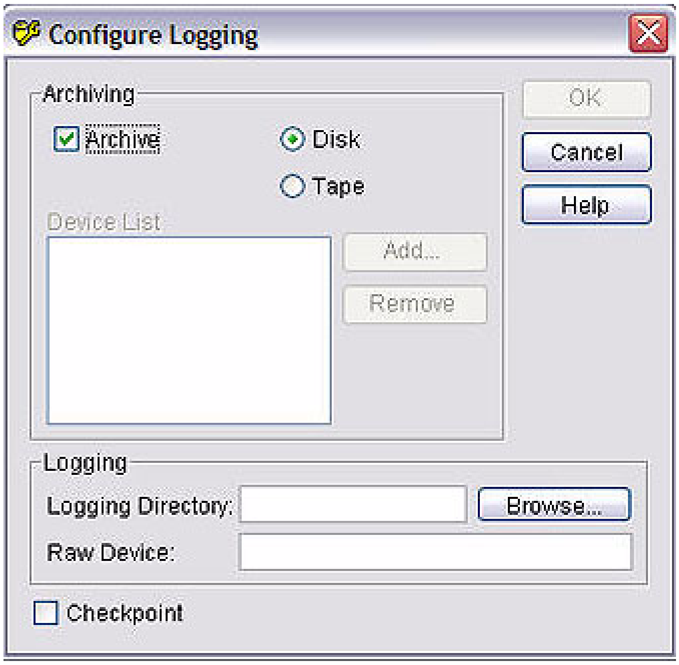
Warning*: If the system crashes before the cache buffer is flushed to tape or the transaction logging system is disabled (which flushes the buffer to tape automatically), some file updates may not be logged to tape.*

When you log updates to tape, one log file is created that spans multiple tapes. When you enable transaction logging, the tape log file is given the next available log file number, which is stored in the LOG.NEXT entry in the dictionary of the UV_LOGS file. To display this number, click Refresh Display on the Transaction Logging window.

Setting Up or Changing the Transaction Logging Configuration

To set up or change the configuration for transaction logging to tape:

1. Choose **Configure -> Logging**. The Configure Logging dialog box appears:



[✓ Archive] [✓ Disk] Tape

2. Select the **Archive** check box. This is the only type of transaction logging available if you are logging to tape.
3. Click the Tape option button

4. Choose one or more tape devices from the Device List.

Initially this list is empty. To select devices to use, click **Add**. The Add Device dialog box appears, which lists all the DC or DT type devices defined in the &DEVICE& file. Select one or more devices from the Tape Devices list and click **OK**. The chosen devices are added to the Device List list. Click **Remove** to remove a chosen tape from the list.

5. Click **OK** to save the settings and to close the dialog box.

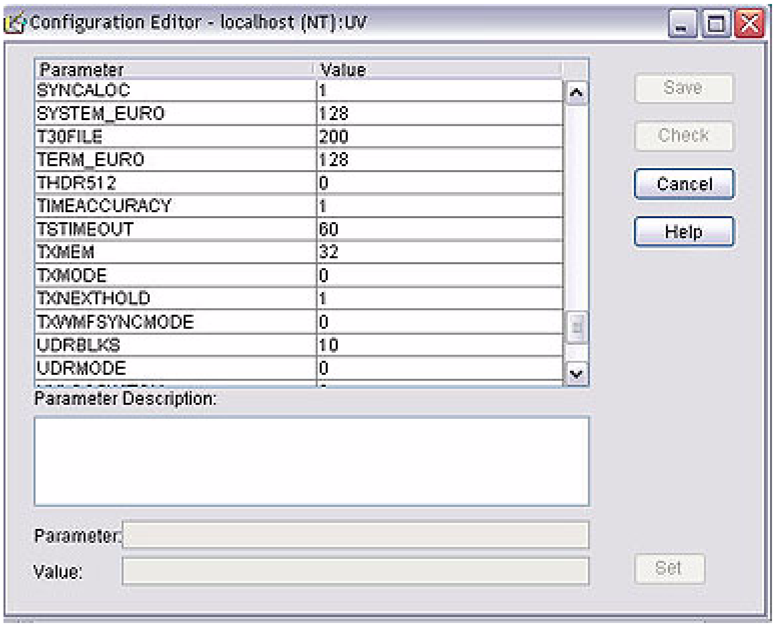
***Note**: To reconfigure transaction logging on a system where it is already enabled, you must shut down transaction logging before you choose the **Logging** option.*

Activating the Transaction Logging System

When you first install UniVerse, the transaction logging system is inactive. To activate the transaction logging system, change the setting of the TXMODE configurable parameter in the *uvconfig* file from 0 (inactive) to 1 (active).

Do the following to change the TXMODE configurable parameter:

1. Click the Configuration Editor icon on the UniAdmin menu. The UniVerse Configuration Editor window appears, displaying a list of parameters.



Parameter	Value
SYNCALOC	1
SYSTEM_EURO	128
T30FILE	200
TERM_EURO	128
THDR512	0
TIMEACCURACY	1
TSTIMEOUT	60
TXMEM	32
TXMODE	0
TXNEXTHOLD	1
TXWMFSYNCMODE	0
UDRBLKS	10
UDRMODE	0

2. Choose TXMODE from the list.
3. Click the Value field and change the TXMODE parameter from 0 to 1.
4. Click Set to accept the new setting.
5. Click Save on the UniVerse Configuration window.

   Note: You cannot use the new setting until you restart UniVerse. You can do this now, or later if it is more convenient.
6. To restart UniVerse on a UNIX system now, choose Yes to run the *DBsetup* program. To restart UniVerse later, choose No, then click OK.

To restart UniVerse now or later on a Windows system, click OK, then manually restart the UniVerse service at your convenience.

Warning: *Before you restart UniVerse, be sure that all users have logged out of UniVerse.*

Changing TXMODE Manually

If you are not using UniAdmin, do the following:

1. Stop all UniVerse processes.
2. Shut down UniVerse.
3. Change your current working directory to the UV account directory.
4. Edit the *uvconfig* file.
5. Set the TXMODE configurable parameter to 1.
6. Execute the UniVerse program *uvregen*.
7. Start up UniVerse.

See *Administering UniVerse* for more information about changing UniVerse configurable parameters, shutting down UniVerse, and starting it up again.

Activating UniVerse Files for Recovery

As part of the setup for transaction logging, you need to choose (activate) which UniVerse files you want to make recoverable. Activated files are referenced in the UV.TRANS file.

***Note**: You also need to configure transaction logging and create log files as part of your setup before you can enable logging.*

On UNIX systems you can activate individual files for recovery, or all files in an account. No global command makes all the files on your system recoverable or nonrecoverable. You can activate only dynamic and static hashed files, type 25 files, and part files that make up distributed files.

Once you activate a file and enable transaction logging, file updates are logged automatically, whether or not the updates are embedded in transactions.

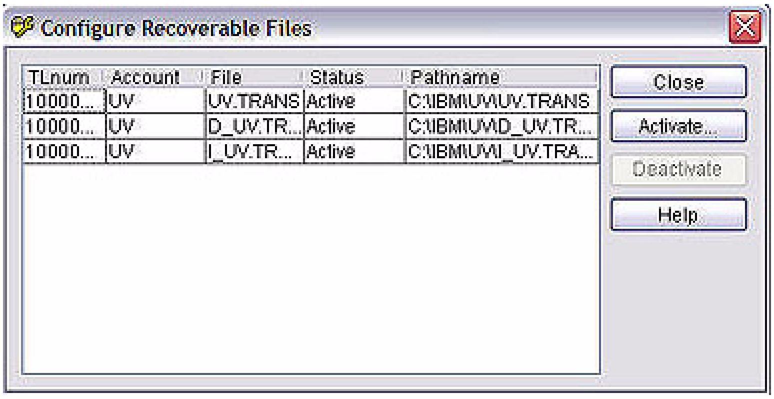
***Note**: Updates to nonrecoverable files in a transaction are not written to the log file and are therefore not recoverable using roll-forward, even when they are included in a transaction.*

You can update a recoverable file outside of a transaction, in which case updates are written individually to the log file. A disadvantage to this is that database integrity may not be preserved.

***Note**: Transaction logging does not allow updates to recoverable files on remote nodes, either inside or outside a transaction.*

Activating Files

To activate files, choose **Configure** ->**Recoverable Files**. The Configure Recoverable Files dialog box appears, as shown in the following example:



TLnum	Account	File	Status	Pathname
10000...	UV	UV.TRANS	Active	C:\IBM\UV\UV.TRANS
10000...	UV	D_UV.TR...	Active	C:\IBM\UV\D_UV.TR...
10000...	UV	I_UV.TR...	Active	C:\IBM\UV\I_UV.TRA...

The Configure Recoverable Files Dialog Box

All the files that have been activated for transaction logging are listed under Active Files. This list includes files that have since been deactivated.

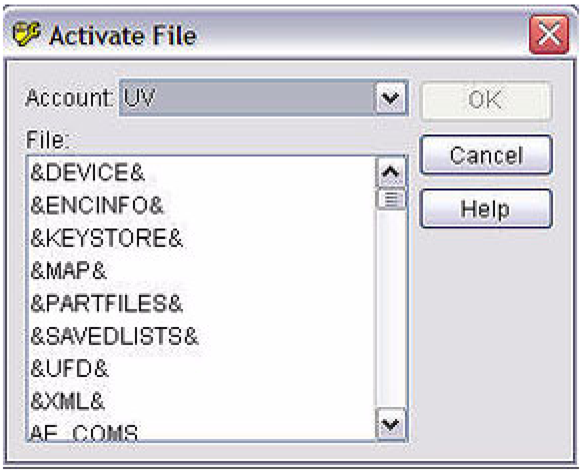
Initially this list is empty. You must activate the files you want to be recoverable. For each activated file, the following information is displayed:

Parameter	**Description**
TLnum	A number assigned to the file when it is activated. This number is the record ID of the corresponding record in the UV.TRANS file.
Account	The UniVerse account where the file is located.
File	The file name of the activated file.
Status	One of two states. Active means the file is still marked as recoverable. An empty status field means the file has been deactivated.
Pathname	The path of the activated file.

Activated File Information

Complete the following steps to activate a file:

1. Click **Activate** to add a file to the Active Files list. The Activate Files dialog box appears, as shown in the following example:



2. Select a UniVerse account from the Account list.
3. Select a file from the Files list. Transaction logging cannot be performed on type 1 or type 19 files. If you select a file of this type, an error message appears. You must acknowledge this message and choose another file.
4. Click **OK**. The Activate Files dialog box closes and the Active Files dialog box is updated with a list of the activated UniVerse files.
5. Click **Close** to close this dialog box.

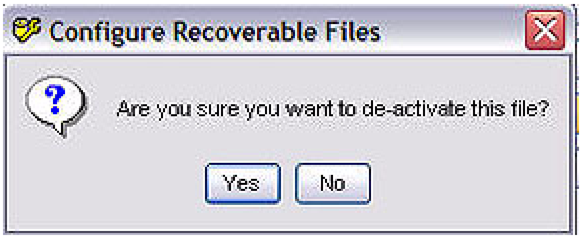
***Note**: You can also activate a deactivated file. To do this, select the file from the Active Files list and click **Activate**. The file's details automatically appear in the Activate Files dialog box. Click **OK**. A message box informs you that this file has been activated on a previous occasion. Click **Yes**. When you activate a deactivated file, any transactions that occurred on that file when it was deactivated are lost.*

Deactivating Files

To deactivate files:

1. Choose **Configure -> Recoverable Files**. The Configure Recoverable Files dialog box appears.
2. Choose the file you want to deactivate in the Active Files list.

3. Click **Deactivate**. The following message box appears.



Are you sure you want to de-activate this file?

4. Click **Yes**. The deactivated file's status is cleared to represent a deactivated file.
5. Click **Close** to close this dialog box, saving the changes you have made.

Note: *Deactivated files remain in the Active Files list. You cannot remove them from the list.*

Making Distributed Files Recoverable

Distributed files are recoverable because they are made up of dynamic files. To make a distributed file recoverable, activate all its part files for transaction logging.

Making Secondary Indexes Recoverable

Updates performed within a transaction are not applied to the UniVerse files until the transaction is committed with the COMMIT statement. Secondary indexes, therefore, do not reflect updates until the COMMIT statement is executed.

When you make a file recoverable by activating it for transaction logging, all secondary indexes belonging to the file are also activated. All updates affecting the file's secondary indexes are logged, including changes made to the index files.

Note: *To guarantee that all updates to files with secondary indexes are logged consistently, all updates must be transactional. Nontransactional updates to a file and its indexes cannot be logged synchronously.*

Adding Indexes to Recoverable Files

If you create a new index for a recoverable file, the new index is not recoverable. To activate the new index, do the following:

1. Suspend or disable transaction logging in one of two ways:
   - Click Suspend or choose Manage -> Suspend Logging.
   - Click Shutdown or choose Manage -> Shutdown Logging.
2. Create new secondary indexes with the CREATE.INDEX command.
3. Back up the file and its indexes to ensure you can recover from media failure.
4. Reactivate the file for transaction logging. Choose Configure -> Recoverable Files -> Activate.
5. Enable transaction logging. Click the Enable button or choose Manage -> Enable Logging.

Deleting Indexes from Recoverable Files

Complete the following steps o delete an index from a recoverable file:

1. Suspend or disable transaction logging. Do either of the following:
   - Click Suspend or choose Manage -> Suspend Logging.
   - Click Shutdown or choose Manage -> Shutdown Logging.
2. Deactivate the file. Choose Configure -> Recoverable Files -> Deactivate.
3. Delete secondary indexes with DELETE.INDEX.
4. Back up the file and its indexes to ensure you can recover from media failure.
5. Activate the file for transaction logging. Choose Configure -> Recoverable Files -> Activate.
6. Enable transaction logging. Click the Enable button or choose Manage -> Enable Logging.

Commands Affecting Recoverable Files

Some UniVerse commands and command options cannot be used on recoverable files because they affect the internal file structure and may destroy the recoverability of the file. Chapter 8, "UniVerse Commands," discusses these commands.

Enabling the Transaction Logging System

To enable transaction logging system-wide, do one of the following:

- Click the **Enable** button.
- Choose **Manage -> Enable Logging**.

It may take some time for the screen to be updated with the enabled status. Click **Refresh Display** to display the new transaction logging state.

Note*: Transaction logging is successfully enabled only if you set the TXMODE configurable parameter to 1 and restart UniVerse. For information about how to change this value, see "Activating the Transaction Logging System" on page 14.*

In the enabled state, updates to recoverable files are written to your UniVerse files and to the current log file.

Suspending Transaction Logging

When transaction logging is suspended, updates to recoverable files are disallowed for the duration of the suspension. To suspend transaction logging system-wide, do one of the following:

- Click the **Suspend** button.
- Choose **Manager -> Suspend Logging**.

Note*: Transaction logging is automatically suspended when all log files are full.*

It may take some time for the screen to be updated with the suspended status. Click **Refresh Display** to display the new transaction logging state.

You must use the **Enable Logging** option to reinstate transaction logging after it has been suspended.

Shutting Down Transaction Logging

If you shut down (disable) transaction logging, updates to recoverable files are written to your UniVerse files, but not to the log files. This means that any changes made to the UniVerse files during the shutdown cannot be recovered.

To shut down transaction logging, do one of the following:

- Click the **Shutdown** button.
- Choose **Manager -> Shutdown Logging**.

It may take some time for the screen to be updated with the disabled status. Click **Refresh Display** to display the new transaction logging state.

You must use the **Enable Logging** option to reinstate transaction logging after a shutdown.

Chapter

Managing Transaction Logging

6

This chapter describes how to manage UniVerse transaction logging. The following topics are discussed:

- Configuring the transaction logging buffer
- Calculating the amount of data you plan to log
- Determining the size and number of log files you need
- Backing up UniVerse files
- Backing up log files to tape
- Releasing and purging backed up log files
- Releasing full tape devices
- Switching between logging to the log directory and logging to a raw device
- Recovering your files
- Managing transaction logging files
- Restarting the transaction logging system

Configuring the Transaction Logging Buffer

The transaction logging buffer is in shared memory. You should configure the size of this buffer to suit your application. This depends on the amount of data logged and how often it is logged (how frequently transactions are committed).

Applications that log only nontransactional data updates risk losing the data in the logging buffer when the system crashes. When transactional updates are committed, they are written to the log file, which reduces the risk of losing data. Unless you specified nonzero values for the LOGSYCNT and LOGSYINT configurable parameters, logged file updates are written synchronously to disk.

***Note**: The changes to the database files themselves are not synchronously written to disk.*

To configure the transaction logging buffer, change the settings of the LOGBLSZ and LOGBLNUM configurable parameters in the *uvconfig* file. The size of the transaction logging buffer is determined by multiplying the value of LOGBLSZ by the value of LOGBLNUM.

- The LOGBLSZ parameter should be the same as the block size of the file system where the log directory is mounted. The default size is 512.
- The LOGBLNUM parameter is the number of blocks in the transaction logging buffer. The default number is 8.

If checkpointing is set to ON, the transaction logging buffer should be at least 16K, because of the larger sizes of warmstart transactions (see Chapter 5, “Setting Up Transaction Logging”). So if the default block size is 512, you should set the number of transaction logging buffer blocks to 32.

If your system is running with NLS enabled, set LOGBLSZ and LOGBLNUM to the following values:

- LOGBLSZ to 2048, or at least to 1024
- LOGBLNUM to 32, or at least to 16

For information about specifying configurable parameter values, see *Administering UniVerse*.

Calculating the Volume of Logged Data

In order to calculate the volume of logged data to be accommodated, you need to estimate the following:

- How many UniVerse files need to be recoverable
- How many updates to those files per hour
- The average size of records in UniVerse files that need to be made recoverable (ANALYZE.FILE can help you determine this)
- How often you fully back up your UniVerse files

You can calculate the volume of logged data as follows:

```
logging.volume = recoverable.updates.per.hour ×
(average.record.size + system.overhead) × hours.between.backups
```

The system overhead varies depending upon the average transaction size and the number of files updated per transaction, but as a general rule, 10% of the average record size is appropriate. You should plan to use a minimum overhead of 40 bytes.

Here is an example. An application generates 18,000 recoverable updates per hour. The average record size is 500 bytes (and a backup of the UniVerse files is performed daily). Substituting these data in the previous formula produces:

```
logging.volume =

18000 recoverable updates per hour
× (500 + 50)(average record size + overhead)
× (1 × 8)hours between backups: 1 day × 8 hours/day

= 80 MB
```

You can reduce the volume of logged data by backing up your UniVerse files more frequently or by reducing the number of recoverable files and, thereby, the number of updates written to the log files.

Determining the Size and Number of Log Files

If you use UniAdmin to create log files, the default log file size is 500K. Using these menus you cannot create a log file smaller than 100K. However you can use the CREATE.LFILE command to create log files smaller than 100K in the log directory. If checkpointing is set to ON, any log file smaller than 100K could produce unpredictable behavior due to the size of warmstart transactions.

Log files need not be the same size. Configure the total logging space to be greater than the amount of space needed to log all log requests occurring between a BEGIN TRANSACTION statement and a COMMIT statement or ROLLBACK statement. This will vary on different systems and is determined by how much data is logged and how long the transaction takes.

To determine the size and number of log files used, consider the following points:

- The minimum log file size is determined by the size of the largest record to be logged. Use the following formula:
- recordID.size + data.size + 40 bytes
- Switching between log files involves extra processing; keeping the number of log files to a minimum reduces this overhead.
- The performance of the commands that manage the log directory depends on the number of log files it contains.
- If your transaction load suddenly increases—for example, during a peak period—you may find your system running short of log file space, forcing you to back up log files to tape.
- You should limit the size of your log files to allow backup and recycling of log file space to be fast enough to deal with emergencies.

Backing Up Your UniVerse Files

If you are using transaction logging, do the following to back up your UniVerse files. It is best to do this when no other users are logged on to the system.

1. Disable transaction logging. Click Shutdown, or choose Manager -> Shutdown Logging from the Transaction Logging window.
2. Back up your UniVerse files and the UV.TRANS and UV_LOGS files.
3. Release any unreleased log files. Click Release, or choose Manager -> Release Log File from the Transaction Logging window.
4. Record the current date and time. This is the starting point in the event of a media failure.
5. Delete obsolete records from the UV.TRANS and UV_LOGS files.
6. Enable transaction logging. Click Enable, or choose Manager -> Enable Logging from the Transaction Logging window.
7. Resume normal operations.

Backing Up Log Files from Disk to Tape

In archive mode, transaction logging to disk requires that you continually back up and release log files between full backups of your UniVerse files to provide a "rolling" transaction logging capability. This complicates day-to-day operations and recovery, but if sufficient disk space is not available to accommodate the log files generated between these full backups, there is no alternative.

How much disk space you have for log files dictates how often you need to back them up. Use the formula for "Calculating the Volume of Logged Data" on page 4 on to determine this frequency.

We recommend that you back up your log files regularly. Once a day or once every two days might be typical. By backing up regularly, you keep the time involved to a predictable minimum.

If possible, back up and release your log files using the appropriate UniAdmin options, because they ensure the safe backup of your log files and the correct updating of the UV_LOGS file.

***Note**: You need not disable or suspend transaction logging while backing up full log files.*

Rolling Forward Raw Device Log Files

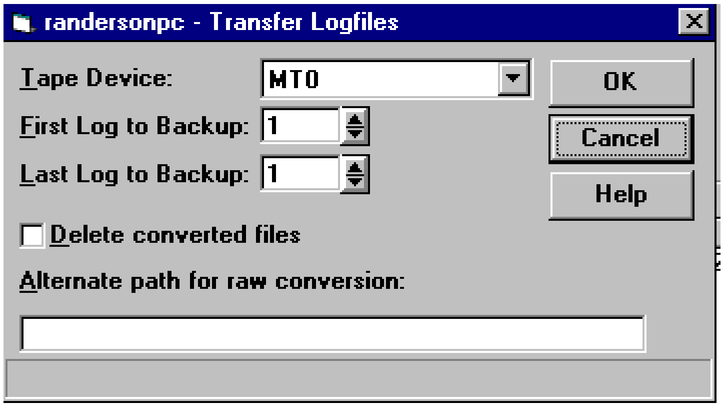
Before you can roll forward log files stored on a raw device, you must first back up and release those log files to tape, then restore them from tape to disk.

Backing Up and Releasing Log Files on Disk or Raw Device

Complete the following steps to back up and release a log file from disk or the raw device to tape:

1. Do one of the following:
   - Click **Transfer**.
   - Choose **Manager -> Transfer Log Files**.

The Transfer Logfiles dialog box appears:



Tape Device: MT0

First Log to Backup: 1

Last Log to Backup: 1

☐ Delete converted files

Alternate path for raw conversion:

1. Select a tape device from the Tape Device list. This list contains all the DC and DT type devices defined in the &DEVICE& file. After you select a device, make sure that a tape is mounted.
2. Specify the log files to transfer in the **First Log to Backup** and **Last Log to Backup** fields. Note that these fields are automatically updated with the numbers of the full log files currently on the system. If you try to enter numbers for log files that do not have a status of Full, an error message appears. You must acknowledge this message and reenter suitable values.
3. (Optional) If you store your log files on a raw device, check the Delete converted files box if you want to delete the backed-up log files after they are released.
4. Do not check this box if you store your log files in the log directory.
5. (Optional) If you store your log files on a raw device, they are moved to the log directory before they are backed up to the chosen device. To move them to a directory other than the log directory, enter the path of an existing directory in the Alternate path for raw conversion field.

   Do not enter a path if you store your log files in the log directory.
6. Click **OK**. The full log files are backed up to the chosen device. Once a check has been made to make sure the backup was successful, the log files are released on disk.

Releasing Log Files on Disk

If a log file on disk is full and you do not need the updates it contains, you can release it without backing it up. This deletes the contents of the log file. The space used by the released file is reallocated to a new log file to maintain the number of available log files.

To keep a copy of the log file contents (in case you want to recover data from the file later), use the **Transfer** option instead.

To release a full log file:

1. Select the full log file from the Log Files list in the Transaction Logging window.
2. Do one of the following:
   - Click **Release**.
   - Choose **Manager -> Release Log File**.

   A message box appears. Click **Yes** to release the log files.

Once you have released a log file, you can remove its entry from the UV_LOGS file using the **Purge** option.

Purging Log Files on Disk

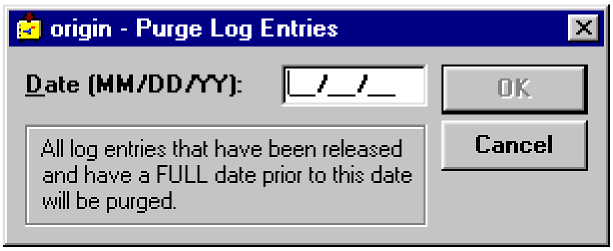
If you transferred or released a full log file on disk, use the **Purge** option to remove its entry from the UV_LOGS file. Regularly purging released log files reduces the number of log files that are searched during data recovery.

Purging is based on a date. Any log files with a Full date before the specified date are purged.

Complete the following steps to purge a log file:

1. Do one of the following:
   - Click **Purge**.
   - Choose **Manager -> Purge Log Files**.

   The Purge Log Entries dialog box appears:



All log entries that have been released and have a FULL date prior to this date will be purged.

2. Enter a date in the **Date (MM/DD/YY)** field.
3. Click **OK**. The entries before the entered date are removed from the UV_LOGS file.

Releasing Full Tape Devices

When you log to tape, you can choose to log to one or more tape devices.

- If you log to a single device, transaction logging is suspended when the tape is full.
- If you log to several devices and a tape is full, logging automatically continues on the next available device. Transaction logging is suspended when all the tapes are full.

In both cases you must mount a new tape and reenable logging. A new log file is created and logging continues.

Transaction logging to tape requires that you continually replace full tapes with new ones, then release the full tape devices to make them available again to the transaction logging system. When a tape device becomes full, do the following:

1. Remove the currently mounted tape from the device, and label it.
2. Mount a new tape to load point.
3. Select the full device from the Tape Device list on the Transaction Logging window.
4. Choose **Manager -> Release Tape**. A message box appears.
5. Click **Yes** to release the tape.

If all tape devices fill up, the transaction logging system enters the full state, and the transaction logging system is suspended. When this happens, do the following:

1. Remove all currently mounted tape media from all full tape devices and label them.
2. Mount new tape media to load point.
3. Enable transaction logging. Click Enable, or choose Manager -> Enable Logging from the Transaction Logging window.

***Warning**: Do not shut down the transaction logging system when all tape devices become full. Shutting down transaction logging clears the tape cache buffer, which could result in losing file updates not yet written to tape.*

Switching the Location of Log Files

You can switch at any time from logging updates in the log directory to logging updates on a raw device, and vice versa. Do the following:

1. Disable transaction logging. Click Shutdown or choose Manager -> Shutdown Logging.
2. Purge all released log files (see "Purging Log Files on Disk" on page 9).
3. Drop all available log files (see Dropping Log Files in Chapter 5, "Setting Up Transaction Logging").
4. Choose **Configure -> Logging…** and do one of the following:
   - If you have been logging to log files in the log directory, in the Raw Device field enter the path of the raw device to which you want to log updates.
   - If you have been logging to log files on a raw device, delete the path of the raw device from the Raw Device field.
5. Click OK.
6. Create new log files in the new log file location (see Adding and Dropping Log Files in Chapter 5, "Setting Up Transaction Logging").

Recovering Application Integrity

Transaction logging helps you maintain some aspects of application integrity by using transaction processing. Events such as power failures can result in partially applied transactions that destroy transactional integrity. Roll-forward recovery restores transactional integrity.

System-wide events, such as a disk drive power failure, may not render disk contents permanently inaccessible. If warmstart recovery is not enabled, you should assume that such events destroy transactional integrity unless you are certain that no update activity has been processed on your system for at least two minutes before the power failure. Warmstart recovery, when enabled, preserves database integrity in such cases.

Single-process events, such as a program that aborts during transaction-commit processing, can also destroy transactional integrity. When this happens, UniVerse sends warning messages to the user's terminal and to the logging information file. If possible, UniVerse automatically suspends recoverable-file updates.

Whenever transactional integrity is destroyed and before reenabling normal system operation, you should restore integrity using roll-forward recovery or some other application-specific programs, or accept the consequences.

Rolling Forward File Updates from Log Files

You can roll forward file updates from log files on disk or from a log file stored on one or more tape devices.

Before you can roll forward file updates from log files on a raw device, you must first transfer the log files to tape, then restore the log files to disk.

Note*: If your system is running with NLS enabled, the log file is marked as an NLS log file. NLS log files can be rolled forward only on a system where NLS is enabled. If a log file is generated on a system with NLS turned off, you cannot roll it forward on a system with NLS turned on.*

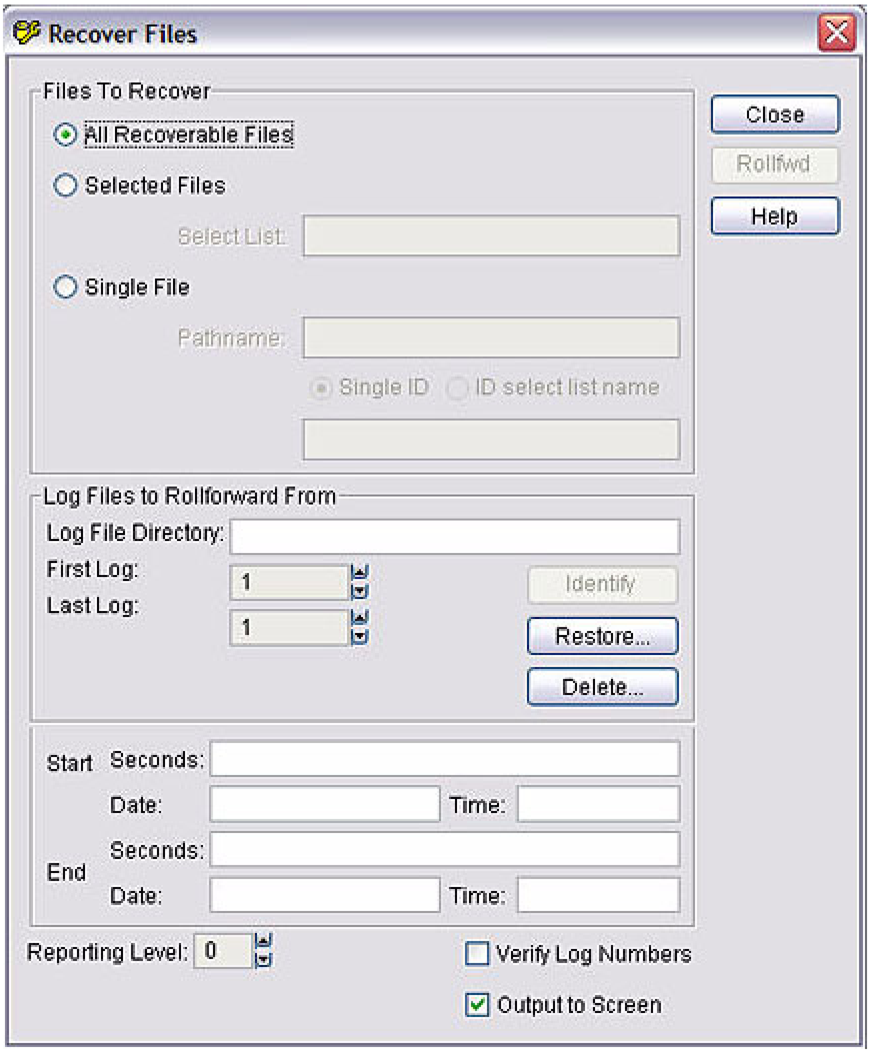
Rolling Forward Log Files from Disk

If the time lapse between the last backup of your UniVerse files and the processing interruption is too large to enable you to restore all log files to disk in a single event, you must repeatedly restore, apply, and delete these files until all have been processed, to recover.

If roll-forward recovery requires multiple restorations of log files, do the following:

1. Suspend or disable transaction logging. Do one of the following:
   - Click Shutdown or choose Manager -> Shutdown Logging.
   - Click Suspend or choose Manage -> Suspend Logging.
2. Restore your UniVerse files (including, if necessary, the UV.TRANS file) from the last full backup. You may need to prepare alternative disks using the operating system's disk management tools.
3. Ensure that the paths in UV.TRANS records match those of the restored files, especially if you use alternative disks.
4. (Optional) If checkpoint mode is set to ON, use the Log Files list on the Transaction Logging window to determine what are the first and last log files you need for roll-forward. The first log file is normally the one that was current when transaction logging was enabled after the last full backup of your UniVerse files.

5. To recover files from a full log file on disk, choose **Recovery -> Rollforward from Disk**. The Recover Files dialog box appears:



Files To Recover
- (●) All Recoverable Files
- ( ) Selected Files
  - Select List:
- ( ) Single File
  - Pathname:
  - (●) Single ID ( ) ID select list name

Log Files to Rollforward From
- Log File Directory:
- First Log: 1
- Last Log: 1

Start Seconds:
Date: Time:
End Seconds:
Date: Time:

Reporting Level: 0

☐ Verify Log Numbers
☑ Output to Screen

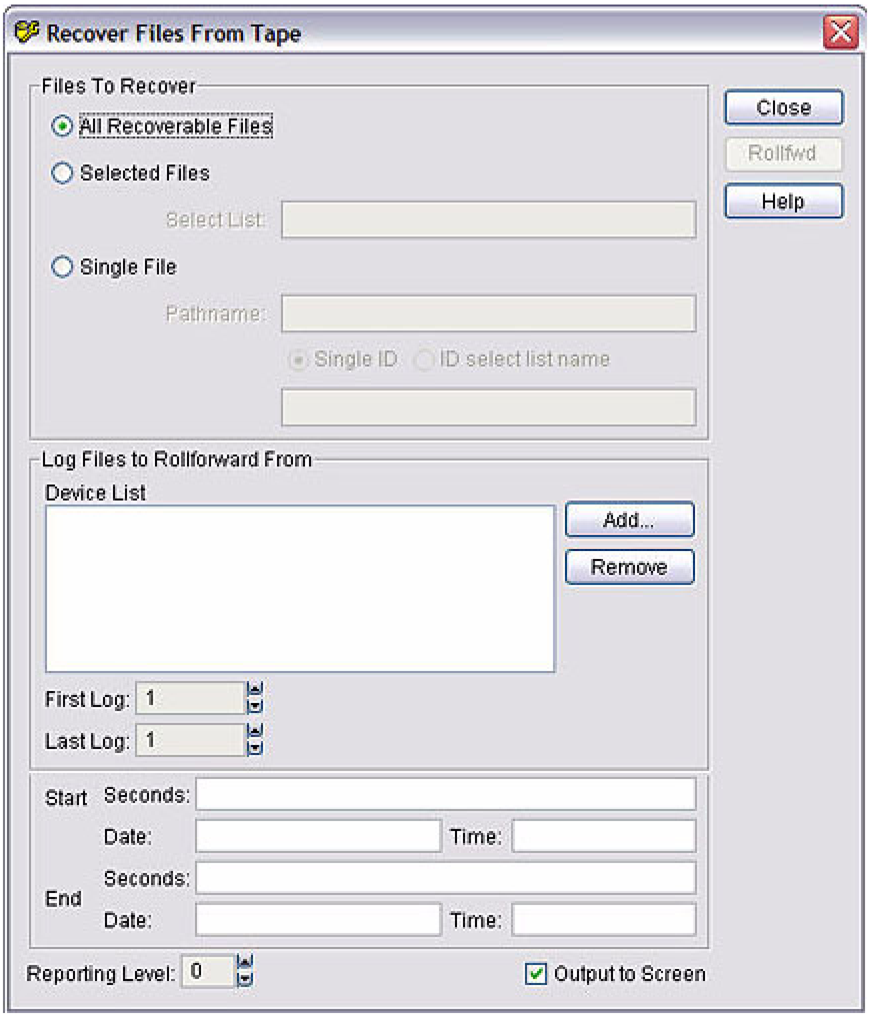
If the log files you need were transferred from disk to tape, you must restore them before specifying the recovery settings.

Restoring Log Files

To recover files from transferred log files, whether transferred from the log directory or from a raw device, you must first restore the log files from tape to disk. When the log files have been restored, you can recover the files you need.

To restore log files from tape:

1. Click **Restore** to transfer the log files from tape to disk. The Restore Logs dialog box appears:



[✓ All Recoverable Files] Selected Files Single File

First Log: 1
Last Log: 1
Reporting Level: 0
[✓ Output to Screen]

2. Select a tape device from the Tape Device list.
3. Choose the log files you want to restore by entering a range of numbers in the **First Log** and **Last Log** fields. You can use the arrows to increase or decrease these values. The default is 1.

4. Choose the directory to restore the log files to by entering the path of a directory in the **Restore To** field. You can also use **Browse** to search the system for a suitable directory.
5. Click **OK**. The chosen log files are restored from tape and are copied to the chosen directory. The Recover Files dialog box remains on the screen so you can recover the files you need.

Recovering Files

Complete the following steps to recover files:

1. Choose the file or files to recover by clicking the appropriate option:

- **All Recoverable Files**. All the activated files are recovered.
- **Selected Files**. A specified named select list is used. The select list must be created in the UV account directory and must contain the paths of the files to be recovered. Enter the name of the select list in the **Select List** box.
- **Single File**. A single file is recovered. Enter the full path of the file in the **Pathname** box.
- **Single Record**. Beginning at UniVerse 10.2, you can define a specific record ID or list or record IDs to recover. To do this, complete the following steps:
    1. Click **Single File**.
    2. In the **Pathname** box, enter the full path to the account where the record ID or list of record IDs exists.
    3. Click **Single ID** if you want to recover one record ID, then enter the ID in the box beneath **Single ID**.
    4. Click **ID select list name** if you want to recover a list of record IDs, then enter the name of the saved list containing the list of IDs from the &SAVEDLISTS& file in the box beneath **ID select list name**.

2. Specify where the log file or files can be found by entering the path in the **Log File Directory** box.
3. Enter the numbers of the first and last log files in the **First Log** and **Last Log** fields. You can use the arrows to increase or decrease these values. The default is 1.

4. If you enabled checkpointing, you can use the **Identify** button. The log file directory is searched and the log files containing the required files are identified. The **First Log** and **Last Log** fields are automatically updated with recommended values. This option is available only if you are restoring all recoverable files or selected files.
5. If you want to recover logs specifying the starting date and time and ending date and time, complete the following steps:
   1. In the **Date/Time** area of the **Recover Files** dialog box, enter the number of seconds from January 1, 1970 at 00:00:00 GMT at which you want to start recovery in the **Start Seconds** box. Do not specify this field if you are defining the Date and Time.
   2. Enter the date on which you want to start recovery in the **Start Date** box. The date must be specified in the yyyy-MM-dd format. Do not enter a date if you specified seconds.
   3. Enter the time at which you want to start recovery in the **Start Time** box. The time must be entered in the HH:mm:ss format. This field is necessary if you are defining the date to start recovery. The default value for the time field is 00:00:00. Do not enter a time if you specified seconds.
   4. Enter the number of seconds from January 1, 1970 at 00:00:00 GMT at which you want to end recovery in the **End Seconds** box. Do not specify this field if you are defining the Date and Time.
   5. Enter the date on which you want to end recovery in the **End Date** box. The date must be entered in the yyyy-MM-dd format. Do not enter a date if you specified seconds.
   6. Enter the time at which you want to end recovery in the **End Time** box. The time must be entered in the HH:mm:ss format. This field is necessary if you are defining the date to end recovery. The default value for the time field is 23:59:59. Do not enter a date if you specified seconds.
6. Choose how much detail to report during the recovery by entering a suitable number in the **Reporting Level** field. You can use the arrows to increase or decrease this value. The minimum setting for this field is 0 (no reporting) and the maximum setting is 3 (highest level of reporting).
7. Select the **Verify Log Numbers** check box if you want the roll-forward program to verify the log numbers during the recovery.

8. Choose where to report the recovery information. If you choose the **Output to Screen** option, all reports appear in the UniVerse Command Output window. This is the default setting. If you uncheck this option, the recovery information is written to the *uvrolf.info* file (the roll-forward information file), which can be viewed (and deleted) at a later date.
9. Click **Rollfwd** to start the recovery. The UniVerse Command Output window appears.
10. When the roll-forward is complete, click **Close** to close the UniVerse Command Output window. You can now delete any restored log files.

Completing the Roll-Forward

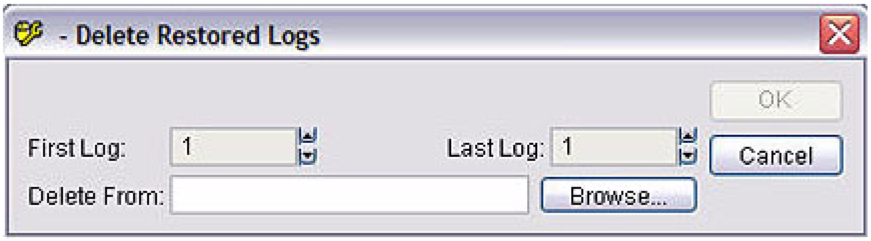
If you need to restore more log files from tape, you are prompted to do so now. Using another window, repeat steps 1 through 3 until all required log files are processed.

1. Choose Recovery -> Rollforward from Disk -> Delete to delete all of the log files just restored and rolled forward to free up disk space.
2. Choose Recovery -> Rollforward from Disk -> Restore… to restore as many of the still required log files as possible.
3. Choose Recovery -> Rollforward from Disk -> Rollfwd to roll forward the log files.
4. (Optional) If possible, make a full backup of all files.
5. Enable transaction logging. Click Enable or choose Manage -> Enable Logging.
6. Resume normal operations.

Deleting Restored Log Files

If you recover files from a transferred log file, you can delete the restored log files as soon as the recovery is complete. To delete restored log files:

1. Click **Delete**. The Delete Restored Logs dialog box appears:



First Log: 1
Last Log: 1
Delete From:

2. Enter the numbers of the log files you want to delete (in a range) in the **First Log** and **Last Log** fields. You can use the arrows to increase or decrease these values.
3. Enter the path of the directory containing the restored log files in the **Delete From** field. You can use **Browse** to search the system for a suitable directory.
4. Click **OK**. The Delete Restored Logs dialog box closes and a message box appears.
5. Click **Yes** to delete the chosen log files.
6. Click **Close** to close the Recover Files dialog box.

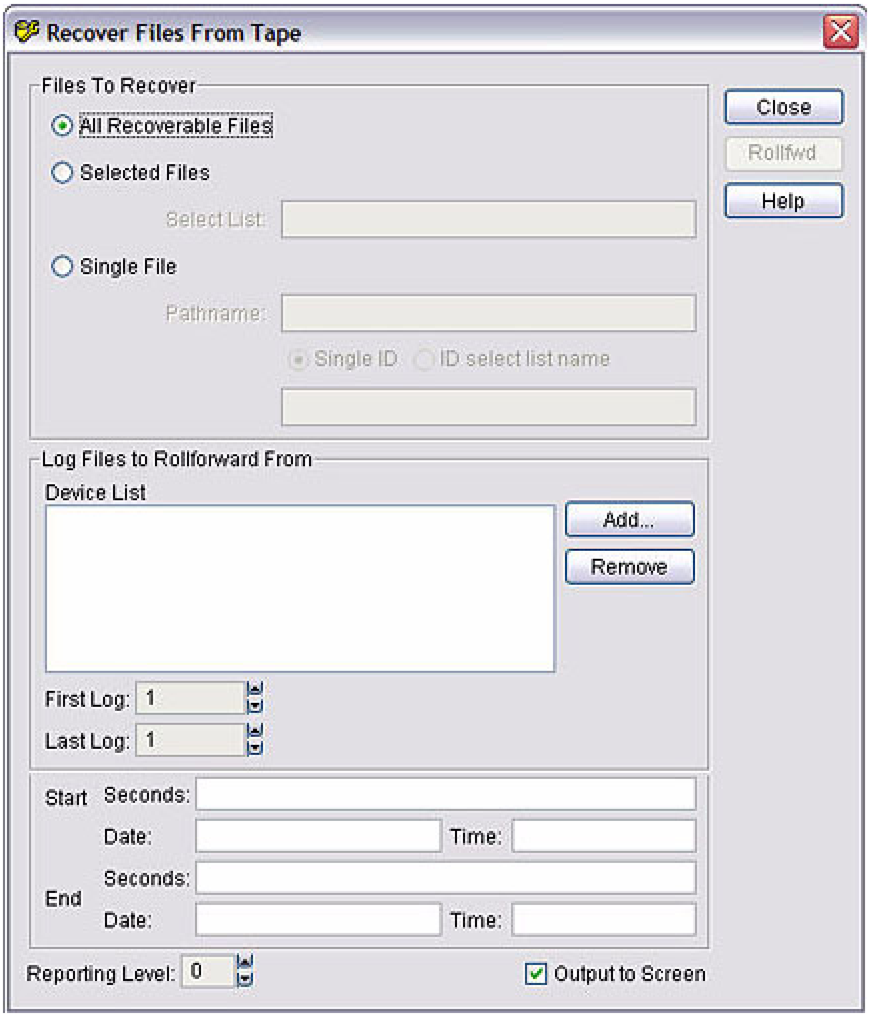
Rolling Forward File Updates from Tape

To roll forward the contents of a log file stored on one or more tape devices, do the following:

1. Suspend or disable transaction logging. Do either of the following:
   - Click Suspend or choose Manage -> Suspend Logging
   - Click Shutdown or choose Manage -> Shutdown Logging
2. Restore your UniVerse files (including, if necessary, the UV.TRANS file) from the last full backup. You may need to prepare alternative disks using the operating system's disk management tools.
3. Ensure that the paths in UV.TRANS records match those of the restored files, especially if you use alternative disks.
4. Mount all the log tapes to load point and in the correct sequence.

   Note: *Since only the first tape in the sequence has a tape label, you must make sure to mount the log tapes in the correct order.*

5. To recover files from a full log file on tape, choose **Recovery Rollforward from Tape**. The Recover Files From Tape dialog box appears:


Recover Files From Tape

Files To Recover
- [● All Recoverable Files]
- ○ Selected Files
  - Select List:
- ○ Single File
  - Pathname:
  - [● Single ID] ○ ID select list name

Log Files to Rollforward From

Device List

First Log: 1

Last Log: 1

Start Seconds:
Date: Time:

End Seconds:
Date: Time:

Reporting Level: 0

[✓ Output to Screen]

6. Choose the file or files to recover by clicking the appropriate option button:
   - **All Recoverable Files**. All the activated files are recovered.
   - **Selected Files**. A specified named select list is used. The select list must be created in the UV account directory and must contain the pathnames of the files to be recovered. Enter the name of the select list in the **Select List** field.
   - **Single File**. A single file is recovered. Enter the full path of the file in the **Pathname** field.

- **Single Record**. Beginning at UniVerse 10.2, you can define a specific record ID or list or record IDs to recover. To do this, complete the following steps:
   7. Click **Single File**.
   8. In the **Pathname** box, enter the full path to the account where the record ID or list of record IDs exists.
   9. Click **Single ID** if you want to recover one record ID, then enter the ID in the box beneath **Single ID**.
   10. Click **ID select list name** if you want to recover a list of record IDs, then enter the name of the saved list containing the list of IDs from the &SAVEDLISTS& file in the box beneath **ID select list name**.

7. Select the device or devices to use from the Device List. Initially this list is empty. To select devices to use, click **Add**. The Add Device dialog box appears, which lists all the DC or DT devices defined in the &DEVICE& file.
8. Select a device or devices from the Tape Devices list and click **OK**. The chosen devices are added to the Device List. Click **Remove** to remove a chosen tape from the list.

   Note: *The devices must be chosen in the order they will be used. If you choose one device and the log file spans more than one tape, you will be prompted to change the tape at the appropriate time. If you use more than one device and the log file spans more than one tape, you must load reel 1 on the first device listed in the Device List.*
9. Enter the numbers of the first and last log files in the **First Log** and **Last Log** fields. You can use the arrows to increase or decrease these values. The default is 1.

10. If you want to recover logs specifying the starting date and time and ending date and time, complete the following steps:
    1. In the **Date/Time** area of the **Recover Files** dialog box, enter the number of seconds from January 1, 1970 at 00:00:00 GMT at which you want to start recovery in the **Start Seconds** box. Do not specify this field if you are defining the Date and Time.
    2. Enter the date on which you want to start recovery in the **Start Date** box. The date must be specified in the yyyy-MM-dd format. Do not enter a date if you specified seconds.
    3. Enter the time at which you want to start recovery in the **Start Time** box. The time must be entered in the HH:mm:ss format. This field is necessary if you are defining the date to start recovery. The default value for the time field is 00:00:00. Do not enter a time if you specified seconds.
    4. Enter the number of seconds from January 1, 1970 at 00:00:00 GMT at which you want to end recovery in the **End Seconds** box. Do not specify this field if you are defining the Date and Time.
    5. Enter the date on which you want to end recovery in the **End Date** box. The date must be entered in the yyyy-MM-dd format. Do not enter a date if you specified seconds.
    6. Enter the time at which you want to end recovery in the **End Time** box. The time must be entered in the HH:mm:ss format. This field is necessary if you are defining the date to end recovery. The default value for the time field is 23:59:59. Do not enter a date if you specified seconds.
11. Choose where to report the recovery information. If you choose **Output to Screen**, all reports appear in the UniVerse Command Output window. This is the default setting. If you uncheck this option, the recovery information is written to the *uvrolf.info* file (the roll-forward information file), which can be viewed (and deleted) at a later date.
12. Choose how much detail to report during the recovery, by entering a suitable number in the **Reporting Level** field. You can use the arrows to increase or decrease the value. The minimum setting for this field is 0 (no reporting) and the maximum setting is 3 (highest level of reporting).
13. Click **Rollfwd** to start the recovery. The UniVerse Command Output window appears.

14. When the roll-forward is complete, click **Close** to close the UniVerse Command Output window.
15. Click **Close** to close the Recover Files dialog box.
16. (Optional) If possible, make a full backup of all files.
17. Enable transaction logging. Click Enable or choose Manage -> Enable Logging.
18. Resume normal operations.

Clearing the File Inconsistency Flag

If a file update fails during warmstart recovery, the file is marked as inconsistent. Once a file has been marked as inconsistent, no further updates to the file are made, and warmstart recovery continues.

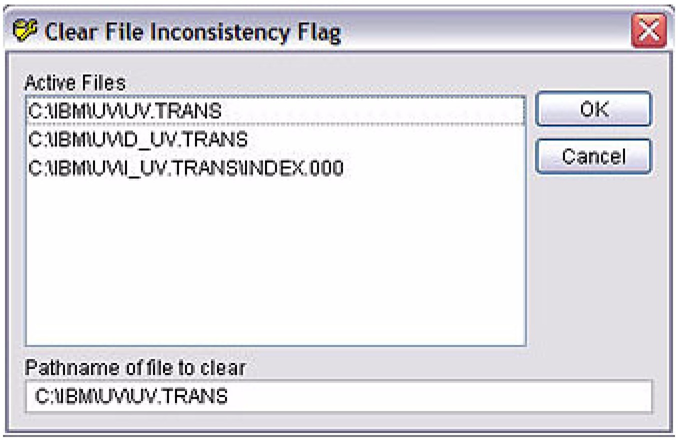
Files updates can fail for a number of reasons. When a file update fails, the *uvrolf.info* file (the roll-forward information file) is updated with the reason for the failure.

To recover data in such cases, you should:

1. Resolve the problem with the file.
2. Clear the inconsistency flag.
3. Roll forward the file from disk or tape.

To clear a file inconsistency flag:

1. Choose **Recovery -> Clear File Inconsistency Flag**. The **Clear File Inconsistency Flag** dialog box appears, as shown in the following example:



C:\IBM\UV\UV.TRANS
C:\IBM\UV\D_UV.TRANS
C:\IBM\UV\I_UV.TRANS\INDEX.000

Pathname of file to clear: C:\IBM\UV\UV.TRANS

2. Choose the inconsistent file, by doing one of the following:
   - Choose the file from the Active Files list. The **Pathname of file to clear** field is updated with the file's path.
   - Enter the path of the file in the **Pathname of file to clear** field.
3. Click **OK**. The file inconsistency flag is removed from the file's header, and updates to the file can take place.

Recovering Accidentally Deleted Files

To recover a file that was accidentally deleted, do the following:

1. Restore the last backup of the deleted file.
2. Recreate, if necessary, the UV.TRANS file record for the file. The record ID *must* be the same as the value originally associated with the deleted file. You may also want to restore the UV.TRANS file itself.
3. Roll forward updates to the file.
   - If you are rolling forward from a log file on disk, choose Recovery -> Rollforward from Disk -> Rollfwd.
   - If you are rolling forward from a log file on tape, choose Recovery -> Rollforward from Tape -> Rollfwd.

Managing Stale Transactions

Stale transactions can occur when transaction logging is running in checkpoint mode only. They can happen when log file space is limited and one or more very large transactions, or transactions that take a long time to complete, fill up all available log files before they are committed or rolled back. They can also occur if a user process is killed while a transaction is active.

The transaction logging system handles stale transactions in two ways. In most cases the stale transactions are aborted, the application fails, and the database is left unchanged. In a few cases, however, the transaction logging system goes into the full state and waits for the administrator to intervene.

When this happens, messages are written to the logging information file (*uvlogd.info*) and the checkpoint information file (*uvchkd.info*). Choose Information Files from the Transaction Logging window, then choose the view option for the file you want to display. If the messages in these files indicate that the transaction logging system is in the full state, you can do either of the following:

- Create more log files, giving the stale transaction space to finish
- Enable transaction logging, aborting all stale transactions

To create more log files and continue, do the following:

1. Create as many new log files as you need to let all stale transactions finish. Click Add or choose Manage -> Add Log Files.
2. Enable transaction logging. Click Enable or choose Manage -> Enable Logging.

To abort any pending stale transactions and continue, do the following:

1. Enable transaction logging. Click Enable or choose Manage -> Enable Logging. The following prompt appears:
2. Terminate pending (prepared) stale transactions?
3. Enter `y` to abort all pending stale transactions.

***Warning**: If the system crashes while the stale transactions are being aborted, a warmstart recovery may leave some of the transactions' updates written to disk.*

Recovering from a UniVerse File Backup Failure

Sometimes a UniVerse file backup fails. When this happens, you can recover your UniVerse files by rolling forward, but at a cost that must be measured against the risk. To recover after a file backup fails, you need the following:

- The UniVerse file backup previous to the corrupted backup you are unable to restore
- A backup of all log files written between the last good UniVerse file backup and the processing interruption

With these requisites, proceed as follows to recover:

1. Restore the last good backup of your UniVerse files (previous to the corrupted one).
2. Restore all log files written between the last good backup and the corrupted UniVerse file backup. Choose Recovery -> Rollforward from Disk from the Transaction Logging window, then click Restore on the Recover Files dialog box.
3. Roll forward the restored log files, using the Recover Files dialog box to roll forward the restored log files.
4. Apply the log files written after the corrupted UniVerse file backup.

In the following example, two successive backups of the UniVerse files are available, as well as a backup of updates written between the backups of these UniVerse files. (Updates written between the latest backup of the UniVerse files and the media failure are also available.) Given these requisites, recovery from media failures remains available despite an inability to restore the latest full backup of the UniVerse files.

Day	Event	Name
1	Full backup of UniVerse files	DB_BU1
2–7	Normal operations: applying updates and writing log files SET1	ALF1

Day	Event	Name
8	Backup of log files SET1	BU_ALF1
	Full backup of UniVerse files	DB_BU2
9–12	Normal operations: applying updates and writing log files SET2	ALF2
13	Media failures	

Given this set of events and the ability to restore the latest backup of the UniVerse files, recovery from media failures involves restoring the UniVerse files from full backup, DB_BU2, and then reapplying the updates from the log files, ALF2.

If you are unable to restore the UniVerse files from full backup DB_BU2, then given the availability of the UniVerse files from full backup, DB_BU1, and the availability of the log files contained in both BU_ALF1 and ALF2, you can still recover from the media failure of Day 13 by doing the following:

1. Restoring the UniVerse files from the previous backup, DB_BU1.
2. If necessary, restoring the log files from BU_ALF1.
3. Applying the updates from BU_ALF1. (Once this is complete, the UniVerse files are effectively the same as those saved on the unrestorable DB_BU2.)
4. Applying the updates from ALF2.
5. Performing a full backup and releasing all log files.
6. Resuming normal operations.

You can use any earlier backup of the UniVerse files as a starting point for transaction logging, provided all log files written before and after that backup are accessible by the roll-forward utility.

Managing Transaction Logging Files

UniVerse transaction logging uses certain system files and directories that must be correctly managed to ensure the success of roll-forward recovery. This section discusses the importance of these files and directories, ways of managing them, and a few cautions associated with some of the management procedures you are likely to develop.

The UV.TRANS File

The UV.TRANS file is essential to the success of the roll-forward utility. UV.TRANS maps unique file identifiers stored in the log files to the paths of the recoverable files to which updates relate. Because of its central importance to the recovery process, you should back up and restore UV.TRANS whenever you back up and restore UniVerse files. Recovery from media failures is impossible if the UV.TRANS file is lost.

Recoverable Files

When a file is made recoverable, a unique file ID is written to the file header. This ID is used as the record ID of the record written to the UV.TRANS file; it is also written to a record in a log file when an update of the recoverable file is logged.

When you roll forward log file updates, the updates are applied only to the original recoverable file according to the mapping information in the UV.TRANS file. Similarly, if a recoverable file is moved from another site or node, the UV.TRANS file has no knowledge of it.

The UV_LOGS File

The roll-forward utility consults the UV_LOGS file when your roll-forward recovery requires that you restore log files from backup. Because all the log files you need may not be available on disk, the UV_LOGS file identifies which log files require restoring.

If the UV_LOGS file is damaged or destroyed, log files that need to be restored may still be identified by referencing an operator or console log. It is more prudent to make sure you back up the UV_LOGS file when you back up the log files.

Log Directory

If a media failure occurs on the disk or partition where the log directory resides, and if that disk or partition is different from that on which your UniVerse files reside, your UniVerse files, while directly unaffected, are rendered vulnerable.

In these circumstances, although the updates contained in the log files have been written to the UniVerse files and the UniVerse files are in an integral state, any subsequent media failures that affect the UniVerse files (in the continued absence of transaction logging) will result in lost updates.

Warning*: If the log files in the log directory are damaged, we recommend that you back up your UniVerse files as soon as possible, as a precaution against subsequent media failures.*

Releasing Log Files

A related issue is the release of backed-up log files. If archive mode is set to ON, these can be released only by the following:

- The RELEASE.LFILE command
- The Transfer Logs dialog box (click Transfer or choose Manage -> Transfer Log Files)
- The Release Log File dialog box (click Release or choose Manage -> Release Log File)

If only checkpoint mode is set to ON, log files are automatically released and reused when they are fully checkpointed.

Both menu options invoke the RELEASE.LFILE command. Only RELEASE.LFILE flags the header of a log file as correctly released. If UniVerse detects the absence of such a marker from a log file, it will not reuse the log file.

If the log directory and the UV_LOGS file are inconsistent, you need to restart the transaction logging system. See "Restarting Transaction Logging" on page 35.

Information Files

The log daemon (*uvlogd*), checkpoint daemon (*uvchkd*), and the roll-forward utility (*uvrolf*) all run as background processes. System and error messages generated by these processes are sent to the following information files in the log directory:

- The logging information file, *uvlogd.info*, which contains messages from the log daemon
- The checkpoint information file, *uvchkd.info*, which contains messages from the checkpoint daemon
- The roll-forward information file, *uvrolf.info*, which contains messages from the roll-forward utility

Viewing an Information File

Choose Information Files from the Transaction Logging window to display the contents of these files. To view an information file, choose one of the following:

- View Logging Information
- View Checkpoint Information
- View Rollforward Information

The chosen file appears in an output window. Click **Close** to close this window.

Note: *If a file is too large to display, a message box appears and only the last 16K bytes of the file are displayed.*

On UNIX systems you can also use the *tail* command with the *–f* option to continuously display updates to these files.

Deleting the Contents of an Information File

To delete the contents of an information file, use the UniAdmin options from the Information Files dialog box. Choose one of the following:

- Delete Logging Information
- Delete Checkpoint Information
- Delete Rollforward Information

When the message box appears, click **Yes**.

Note: *Be careful not to delete the roll-forward information file while a roll-forward is in progress, because you will lose all further output from the roll-forward. Likewise, do not delete the checkpoint information file while the checkpoint daemon is active. And do not delete the logging information file while transaction logging is in the initializing, warmstart, or enabled state.*

Restarting Transaction Logging

This section describes the procedures to use when you need to restart the transaction logging system. Two reasons you may need to do this are as follows:

- Recoverable files were updated while transaction logging is disabled.
- Transaction logging files and the UV_LOGS file are inconsistent.

Restarting After Disabling Transaction Logging

Sometimes updates are made to recoverable files after you disable transaction logging. Because these updates are not logged, you cannot roll them forward to restore your files to the most recent integral state. To fully reinstate the transaction logging system after you have disabled it, do the following:

1. Stop all UniVerse processes.
2. Back up your UniVerse files.
3. Enable transaction logging. Click Enable or choose Manage -> Enable Logging.

Restarting Transaction Logging from the Beginning

Sometimes you need to restart UniVerse transaction logging from the beginning. For example, if the state of the log files in the log directory and the UV_LOGS file are inconsistent, you must restore their consistency before you reenable transaction logging. To restart transaction logging, do the following:

1. Disable transaction logging. Click Shutdown or choose Manage -> Shutdown Logging.
2. Check the Log Files list to see if there are any Full log files.
   - If archive mode is set to ON, click Transfer or choose Manage -> Transfer Log Files.
   - If archive mode is set to OFF, click Release or choose Manage -> Release Log File.
3. Delete all Available log files from the log directory. Click Drop or choose Manage -> Drop Log Files.

4. Remove all out-of-date records from the UV_LOGS file that refer to released log files. Click Purge or choose Manage -> Purge Log Files.
5. Check the Log Files list to see if there are any Current or NeedsSync log files. If you need these log files to recover from a media failure, copy them to another directory.

   ***Note**: After copying these files, the roll-forward utility can no longer use them for a warmstart recovery.*
6. Change directory to the log directory and remove all log files from the log directory.
7. Use the Log Files list to find the number of the Current log file, then edit the LOG.NEXT record in the dictionary of UV_LOGS to be one more than that. For example:

```
>ED DICT UV_LOGS LOG.NEXT
2 lines long.

----: 2
0002: 24
----: R 25
0002: 25
----: FI
```

8. Delete any records in UV_LOGS with record IDs equal to or higher than the log file number in the LOG.NEXT record. For example:

```
>DELETE UV_LOGS 25 26 27
```

9. Create new log files. Click Add or choose Manage -> Add Log Files.
10. Shut down UniVerse and restart it to clear shared memory.
11. Enable UniVerse transaction logging. Click Enable or choose Manage -> Enable Logging.

Chapter

Modifying Applications

7

This chapter summarizes the main things to consider when you are deciding what modifications you need to make to existing applications to use transaction logging. The following topics are discussed:

- Determining which UniVerse files you should make recoverable
- Identifying modifications required for using transaction processing

Determining Which Files to Make Recoverable

In determining which of your UniVerse files to make recoverable, consider the following questions:

- For the purposes of transaction logging, what files should be deemed related? A bank, for example, may regard the ability to recover its Customers file *and* its Accounts file, as equally imperative.
- How important is the data in the file? Implicit in such a question are further questions such as:
  - Can you re-create the file if it is lost?
  - If the file can be re-created, how long will its re-creation take?
  - What will be the cost in resources of re-creating the file?
  - What will be the cost of being unable to re-create the file?
- How frequently is the file likely to be updated? (This may bear no relation to the file's size.)
- What is the cost in system performance of making a specified file or group of files recoverable?
- How often must you back up your UniVerse files if you make a specified group of files recoverable?
- Given a desired frequency of backup of your UniVerse files, will your available disk space accommodate the volume of logged updates generated by the files you want to make recoverable?

Program Changes to Consider

The following model of a typical program provides a useful framework for discussing potential modifications you might make to your existing applications when you use transaction logging:

1. Input/Verification. A banking program, for example, prompts for the Account Number of the customer whose account is to be updated and then verifies the existence of that input Account Number.
2. Locking/Reading. The same banking program locks the record specified by the input Account Number and displays the contents of the record at the user's terminal.
3. Updating. The banking program allows the user to update the record specified by the Account Number. When updating is successfully completed, record locks are released.

Input/Verification

Because transactional updates to recoverable files require the maintenance of update record locks on the records being updated, and given the competition for access to these records, your programs should maintain such locks as briefly as possible. For this reason, your programs should not ask for user input during transactions. Programs asking for user input during transactions proceed at the pace of a user's convenience and run the risk of maintaining locks for an indefinite period.

Locking/Reading

Make sure your program locks a record before reading and updating it.

Updating

To make full use of transaction logging, update recoverable files within transactions. Transaction processing requires that you delimit a set of logically related updates to recoverable files with the UniVerse BASIC transaction statements BEGIN TRANSACTION statement, and COMMIT statement or ROLLBACK statement.

Because record locks set for a transaction are released by the execution of a COMMIT or ROLLBACK statement (and should be released only by these statements), remove any RELEASE statements from within transactions, and check for redundant RELEASE statements following transactions.

***Note**: When a program executes the COMMIT statement, its deferred updates are applied to the appropriate files, then all locks are released. Alternatively, if a program executes the ROLLBACK statement, its deferred updates are discarded, then all locks set by the program are released.*

The application usually determines the number of updates in a transaction. There is, however, some performance benefit by incurring overhead for only one transaction with many small updates. The performance benefit must be traded off against possible increased wait times and degraded throughput resulting from record locks and (especially) file locks being held for longer periods. Records updated by the transaction remain locked until the COMMIT or ROLLBACK statement is executed. Also, minimize the amount of general processing done in transactions.

Try to ensure that transactions are not excessively large or will not take an excessively long time to process. It may be necessary to divide a set of logically related updates that exceeds total log file capacity into logically related subsets, although it is your responsibility in such circumstances to monitor these subsets.

Chapter

UniVerse Commands

8

This chapter describes two groups of UniVerse commands:

- Commands used for transaction logging
- Commands you cannot use with recoverable files

The following UniVerse commands are described:

- ACTLIST
- CREATE.LDIR
- CREATE.LFILE
- DEACTLIST
- DEL.RFILE
- DELETE.LFILE
- ENABLE.RECOVERY
- LOG.SAVE
- LOG.RESTORE
- MKFILELIST
- RECOVERY.CHECKPOINT
- RECOVERY.CONSISTENT
- RELEASE.LFILE
- SET.LOG.ATTR
- SHUTDOWN.RECOVERY
- SUSPEND.RECOVERY

Most of these commands can be used only by UniVerse administrators.

Activating Recoverable Files

Use the following commands to activate and deactivate lists of recoverable files:

- MKFILELIST creates a select list of all files in an account and saves it in the &SAVEDLISTS& file. Each element in the list contains the account name and a file name. You can edit this list to include only those files you want to make recoverable.
- ACTLIST activates for recovery all files listed in a saved select list created by MKFILELIST.
- DEACTLIST deactivates all files listed in a select list.

Enabling and Disabling Transaction Logging

Use the following commands to enable, disable, and suspend the transaction logging system:

- ENABLE.RECOVERY starts up the log daemon.
- SHUTDOWN.RECOVERY shuts down the log daemon.
- SUSPEND.RECOVERY suspends the log daemon.

To enable warmstart transaction logging, use SET.LOG.ATTR to set checkpoint mode to ON. To enable media recovery, use SET.LOG.ATTR to set archive mode to ON. You must run the transaction logging system in one of these modes. You can run it in both.

Creating Log Files

Use the following commands to create transaction logging system log files and the directory where they are stored:

- CREATE.LDIR creates the log directory.
- CREATE.LFILE creates one or more log files in the log directory.
- DELETE.LFILE deletes empty log files whose status is Available from the log directory.

Managing Log Files

Use the following commands when you are releasing Full log files for reuse, backing up or restoring log files from tape to disk, rolling forward the restored log files, and deleting log files once you have rolled them forward:

- RELEASE.LFILE releases a Full log file for reuse.
- LOG.SAVE saves log files from a log directory on disk to tape.
- LOG.RESTORE restores log files from tape to a log directory on disk.
- RECOVERY.CHECKPOINT finds the numbers of the first log file you need for a roll-forward recovery.
- DEL.RFILE deletes a series of log files you have rolled forward, freeing up disk space.
- RECOVERY.CONSISTENT clears a file's inconsistency flag.

Prohibited UniVerse Commands

You cannot use the following UniVerse commands with recoverable files, because they alter the internal structure of files over time. Such alteration would cause roll-forward recovery to function incorrectly.

- CONFIGURE.FILE
- CREATE.INDEX
- DELETE.INDEX
- DEFINE.DF

If you deactivate files for recovery so you can use the prohibited commands, you must back them up before you reactivate them for recovery. This will ensure that roll-forward will work properly.

Index