



Teradata Archive/Recovery Utility

Reference

Release 12.00.00

B035-2412-067A

July 2007

The product or products described in this book are licensed products of Teradata Corporation or its affiliates.

Teradata, BYNET, DBC/1012, DecisionCast, DecisionFlow, DecisionPoint, Eye logo design, InfoWise, Meta Warehouse, MyCommerce, SeeChain, SeeCommerce, SeeRisk, Teradata Decision Experts, Teradata Source Experts, WebAnalyst, and You've Never Seen Your Business Like This Before are trademarks or registered trademarks of Teradata Corporation or its affiliates.

Adaptec and SCSISelect are trademarks or registered trademarks of Adaptec, Inc.

AMD Opteron and Opteron are trademarks of Advanced Micro Devices, Inc.

BakBone and NetVault are trademarks or registered trademarks of BakBone Software, Inc.

EMC, PowerPath, SRDF, and Symmetrix are registered trademarks of EMC Corporation.

GoldenGate is a trademark of GoldenGate Software, Inc.

Hewlett-Packard and HP are registered trademarks of Hewlett-Packard Company.

Intel, Pentium, and XEON are registered trademarks of Intel Corporation.

IBM, CICS, DB2, MVS, RACF, Tivoli, and VM are registered trademarks of International Business Machines Corporation.

Linux is a registered trademark of Linus Torvalds.

LSI and Engenio are registered trademarks of LSI Corporation.

Microsoft, Active Directory, Windows, Windows NT, and Windows Server are registered trademarks of Microsoft Corporation in the United States and other countries.

Novell and SUSE are registered trademarks of Novell, Inc., in the United States and other countries.

QLogic and SANbox trademarks or registered trademarks of QLogic Corporation.

SAS and SAS/C are trademarks or registered trademarks of SAS Institute Inc.

SPARC is a registered trademarks of SPARC International, Inc.

Sun Microsystems, Solaris, Sun, and Sun Java are trademarks or registered trademarks of Sun Microsystems, Inc., in the United States and other countries.

Symantec, NetBackup, and VERITAS are trademarks or registered trademarks of Symantec Corporation or its affiliates in the United States and other countries.

Unicode is a collective membership mark and a service mark of Unicode, Inc.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other product and company names mentioned herein may be the trademarks of their respective owners.

THE INFORMATION CONTAINED IN THIS DOCUMENT IS PROVIDED ON AN “AS-IS” BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSION MAY NOT APPLY TO YOU. IN NO EVENT WILL TERADATA CORPORATION BE LIABLE FOR ANY INDIRECT, DIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS OR LOST SAVINGS, EVEN IF EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The information contained in this document may contain references or cross-references to features, functions, products, or services that are not announced or available in your country. Such references do not imply that Teradata Corporation intends to announce such features, functions, products, or services in your country. Please consult your local Teradata Corporation representative for those features, functions, products, or services available in your country.

Information contained in this document may contain technical inaccuracies or typographical errors. Information may be changed or updated without notice. Teradata Corporation may also make improvements or changes in the products or services described in this information at any time without notice.

To maintain the quality of our products and services, we would like your comments on the accuracy, clarity, organization, and value of this document. Please e-mail: teradata-books@lists.teradata.com

Any comments or materials (collectively referred to as “Feedback”) sent to Teradata Corporation will be deemed non-confidential. Teradata Corporation will have no obligation of any kind with respect to Feedback and will be free to use, reproduce, disclose, exhibit, display, transform, create derivative works of, and distribute the Feedback and derivative works thereof without limitation on a royalty-free basis. Further, Teradata Corporation will be free to use any ideas, concepts, know-how, or techniques contained in such Feedback for any purpose whatsoever, including developing, manufacturing, or marketing products or services incorporating Feedback.

Copyright © 1997-2007 by Teradata Corporation. All Rights Reserved.

Preface

Purpose

This book provides information about Teradata Archive/Recovery Utility (Teradata ARC), which is a Teradata® Tools and Utilities product. Teradata Tools and Utilities is a group of products designed to work with Teradata Database.

Teradata ARC writes and reads sequential files on a Teradata client system to archive, restore, recover, and copy Teradata Database table data. Through its associated script language, it also provides an interface between Teradata's Open Teradata Backup (OTB) solutions and the Teradata Database.

Audience

This book is intended for use by:

- System administrators
- Database administrators
- Other technical personnel responsible for maintaining a Teradata Database

Supported Releases

This book supports the following releases:

- Teradata Database 12.00.00
- Teradata Tools and Utilities 12.00.00
- Teradata ARC 12.00.00

Note: See “[Verifying Teradata ARC Version Number](#)” on page 15 to verify the Teradata ARC version number.

To locate detailed supported-release information:

- 1 Go to www.info.teradata.com.
- 2 Navigate to **General Search>Publication Product ID**.
- 3 Enter 3119.
- 4 Open the version of the *Teradata Tools and Utilities ###.##.## Supported Versions* spreadsheet associated with this release.

The spreadsheet includes supported Teradata Database versions, platforms, and product release numbers.

Prerequisites

The following prerequisite knowledge is required for this product:

- Basic computer technology
- NCR system hardware
- Teradata Database
- System console environment
- X Windows

Changes to This Book

The following changes were made to this book in support of the current release. Changes are marked with change bars. For a complete list of changes to the product, see the *Release Definition* associated with this release.

Date and Release	Description
July 2007 12.00.00	<ul style="list-style-type: none">• Added online archive logging functionality with LOGGING ONLINE ARCHIVE ON and LOGGING ONLINE ARCHIVE OFF.• Removed support for ASF2 Tape Reader.• Added DBC.Dependency to Table 3 on page 29.• Clarified concurrent use of locks during archive.• Clarified that Teradata ARC locks are HUT locks.• Added DBSERROR command line parameter.• Added note that not all DBC tables are archived.• Added ONLINE and KEEP LOGGING options to ARCHIVE.• Clarified that MP-RAS and Linux support exists for IOPARM and IOMODULE.

Additional Information

Additional information that supports this product and Teradata Tools and Utilities is available at the web sites listed in the table that follows. In the table, *mmyx* represents the publication date of a manual, where *mm* is the month, *y* is the last digit of the year, and *x* is an internal publication code. Match the *mmy* of a related publication to the date on the cover of this book. This ensures that the publication selected supports the same release.

Type of Information	Description	Access to Information
Release overview Late information	<p>Use the Release Definition for the following information:</p> <ul style="list-style-type: none"> • Overview of all of the products in the release • Information received too late to be included in the manuals • Operating systems and Teradata Database versions that are certified to work with each product • Version numbers of each product and the documentation for each product • Information about available training and the support center 	<ol style="list-style-type: none"> 1 Go to http://www.info.teradata.com/. 2 Select the General Search check box. 3 In the Publication Product ID box, type <i>2029</i>. 4 Click Search. 5 Select the appropriate Release Definition from the search results.
Additional product information	<p>Use the Teradata Information Products Publishing Library site to view or download specific manuals that supply related or additional information to this manual.</p>	<ol style="list-style-type: none"> 1 Go to http://www.info.teradata.com/. 2 Select the Teradata Data Warehousing check box. 3 Do one of the following: <ul style="list-style-type: none"> • For a list of Teradata Tools and Utilities documents, click Teradata Tools and Utilities and then select a release or a specific title. • Select a link to any of the data warehousing publications categories listed. <p>Specific books related to Teradata ARC are as follows:</p> <ul style="list-style-type: none"> • <i>Data Dictionary</i> B035-1092-mmyx • <i>Messages</i> B035-1096-mmyx • <i>Open Teradata Backup Release Definition</i> B035-3114-mmyx • <i>Teradata Call-Level Interface Version 2 Reference for Network-Attached Systems</i> B035-2418-mmyx • <i>Teradata Tools and Utilities Installation Guide for Microsoft Windows</i> B035-2407-mmyx • <i>Teradata Tools and Utilities Installation Guide for IBM z/OS</i> B035-2458-mmyx • <i>Teradata Tools and Utilities Installation Guide for IBM VM</i> B035-2422-mmyx • <i>Teradata Tools and Utilities Installation Guide for UNIX and Linux</i> B035-2459-mmyx

Type of Information	Description	Access to Information
CD-ROM images	Access a link to a downloadable CD-ROM image of all customer documentation for this release. Customers are authorized to create CD-ROMs for their use from this image.	<ol style="list-style-type: none">1 Go to http://www.info.teradata.com/.2 Select the General Search check box.3 In the Title or Keyword box, type <i>CD-ROM</i>.4 Click Search.
Ordering information for manuals	Use the Teradata Information Products Publishing Library site to order printed versions of manuals.	<ol style="list-style-type: none">1 Go to http://www.info.teradata.com/.2 Select the How to Order check box under Print & CD Publications.3 Follow the ordering instructions.
General information about Teradata	<p>The Teradata home page provides links to numerous sources of information about Teradata. Links include:</p> <ul style="list-style-type: none">• Executive reports, case studies of customer experiences with Teradata, and thought leadership• Technical information, solutions, and expert advice• Press releases, mentions, and media resources	<ol style="list-style-type: none">1 Go to Teradata.com.2 Select a link.

Table of Contents

Preface	3
Purpose	3
Audience	3
Supported Releases	3
Prerequisites	4
Changes to This Book	4
Additional Information	4

Chapter 1: Introduction	15
What is Teradata ARC?	15
Platform Support	15
Verifying Teradata ARC Version Number	15
Uses of Teradata ARC	17
What is ARCMAN?	17
Starting ARCMAN	17
Starting ARCMAN from MVS	18
Starting ARCMAN from VM	19
Starting ARCMAN from Linux, MP-RAS, and Windows 2000/XP/Server 2003	20
Canceling Teradata ARC	22
Sample Teradata ARC Scripts	22
Data Extension Modules	23

Chapter 2: Archive/Recovery Operations	25
Database DBC	25
Restoring	27
Database SYSUDTLIB	27
Archiving	27
Restoring	27

Copying.....	27
Deleting.....	28
Archiving Tables and Databases	28
Concurrency Control.....	29
Archiving With All AMPs Online.....	29
Archiving by Cluster	35
Archiving With Offline AMPs	36
Archiving Large Objects (LOBs).....	37
Archiving Non-Hashed and Partially Loaded Tables	37
Encrypting Archived Data.....	37
Archiving Online.....	38
Determining Whether Online Archive Logging is in Use	38
Restoring Tables and Databases.....	39
Conditions Needed for a Restore Operation.....	39
Considerations Before Restoring Data.....	40
Restoring the Database DBC.....	41
Restoring a User Database or Table	42
Restoring With All AMPs Online.....	46
Restoring with a Specific-AMP Archive.....	46
Restoring Using the EXCLUDE Option	47
Restoring With AMPs Offline.....	48
Restoring With One AMP Offline	48
Restoring Cluster Archives	49
Restoring After a Reconfiguration	51
Restoring with a Larger Number of AMPs	51
Restoring a Specific AMP	52
Restoring Encrypted Data from an Archive File.....	52
Recovering Tables and Databases	52
Recovering With Offline AMPs	53
Recovering a Specific AMP	54
Copying Tables and Databases	54
Conditions for Using the COPY Statement.....	55
COPY Examples.....	56
Copying Encrypted Data from an Archive File.....	59
Using Host Utility Locks	60
Teradata ARC Locks During an Archive Operation.....	61
Teradata ARC Locks During a Restore Operation	63
Locks Associated with Other Operations.....	64
Setting Up Journal Tables	64
Location of Change Data.....	65
Local Journaling.....	67
Archiving Journal Tables.....	67

Journal Impact on Recovery	67
Controlling Journal Checkpoint Operations	68
Checkpoint Names.....	68
Submitting a CHECKPOINT Statement	68
Checkpoint and Locks	68
Completing a Checkpoint With Offline AMPs.....	69

Chapter 3:

Environment Variables..... 71

ARCDFLT	72
ARCENV and ARCENVX	73

Chapter 4:

Runtime Parameters..... 75

CATALOG.....	78
CHARSETNAME	83
CHECKPOINT	88
CHECKSUM	90
DATAENCRYPTION.....	91
DBSERROR.....	92
DEFAULT	93
DEMODULE.....	95
DEPARM.....	97
ERRLOG	99
FATAL	100
FILEDEF	101
HALT	103
HEX	104
IOMODULE	105
IOPARM	106
LOGON	107
LOGSKIPPED	108
OUTLOG.....	109
PARM.....	110
PAUSE	113

PERFFILE	114
RESTART.....	116
RESTARTLOG	117
SESSIONS	118
STARTAMP.....	120
UEN (Utility Event Number)	121
VERBOSE	122
WORKDIR	126

Chapter 5:

Return Codes and UNIX Signals

Return Codes.....	127
UNIX Signals.....	132

Chapter 6:

Archive/Recovery Control Language

ANALYZE	135
ARCHIVE	137
BUILD	150
CHECKPOINT	152
COPY.....	155
DELETE DATABASE	168
DELETE JOURNAL	170
ENABLE DATA EXTENSION.....	172
ENABLE ENCRYPTION	174
LOGDATA.....	176
LOGGING ONLINE ARCHIVE OFF.....	177
LOGGING ONLINE ARCHIVE ON	179
LOGMECH	181
LOGOFF	182
LOGON	183
RELEASE LOCK	185
RESTORE.....	188
REVALIDATE REFERENCES FOR	202
ROLLBACK.....	205

ROLLFORWARD	208
SET QUERY_BAND	212

Chapter 7: Restarting Teradata ARC

Restart Log	213
Restarting Teradata ARC	213
During a Database DBC Operation	213
During an Archive Operation	214
During a Restore Operation	214
During a Recovery Operation	214
Restart After Client or Teradata ARC Failure	214
Restarting an Archive Operation	215
Restarting a Restore Operation	216
Restarting a Checkpoint Operation	217
Restart After a Teradata Database Failure	217
Restarting an Archive Operation	217
Restarting a Restore Operation	218
Restarting a Recovery Operation	218
Restarting a Checkpoint Operation	218
Removing HUT Locks After a Restart	218
Recovery Control Catalog	219
Recording Row Activity	219
Recording AMP Information	221
Recording Device Information	222

Appendix A: How to Read Syntax Diagrams

Syntax Diagram Conventions	223
----------------------------------	-----

Appendix B: Multivolume CMS Tape Support

CMS Tape Support Messages	229
---------------------------------	-----

Glossary231

Index.....235

List of Tables

Table 1: Minimum Region Sizes Running from MVS.....	19
Table 2: Tables Archived for Database DBC.....	25
Table 3: Dictionary Rows Archived for User Database.....	29
Table 4: User Database Data Dictionary Rows Restored.....	45
Table 5: User Table Dictionary Rows Restored	45
Table 6: Journal Options	65
Table 7: Journal Change Row Location	66
Table 8: Environment Variables	71
Table 9: Runtime Parameters.....	75
Table 10: Teradata-Defined Character Sets	84
Table 11: Data Session Requirements	119
Table 12: Return Codes	128
Table 13: Database Error Messages	129
Table 14: Client-Generated Error Messages.....	130
Table 15: Summary of Teradata ARC Statements	133
Table 16: Messages Routed to System Operator	229

This chapter contains these topics:

- [What is Teradata ARC?](#)
- [What is ARCMAN?](#)
- [Starting ARCMAN](#)
- [Canceling Teradata ARC](#)
- [Sample Teradata ARC Scripts](#)
- [Data Extension Modules](#)

What is Teradata ARC?

Teradata ARC writes and reads sequential files on a Teradata client system to archive, restore, and recover, as well as to copy, Teradata Database table data.

Platform Support

- IBM MVS/VM
- Linux RedHat/SUSE
- NCR UNIX MP-RAS
- Windows 2000/XP
- Windows Server 2003

Verifying Teradata ARC Version Number

The version number for Teradata ARC is always displayed in the startup banner in Teradata ARC output. For example:

```
03/16/2007 10:06:19 Copyright 1989-2007, NCR Corporation.
03/16/2007 10:06:20 All Rights Reserved.
03/16/2007 10:06:20
03/16/2007 10:06:20      ***      *****      *****
03/16/2007 10:06:20      *      *      *      *      *      PROGRAM: ARCMAN
03/16/2007 10:06:20      *****      *****      *      RELEASE: 12.00.00.00
03/16/2007 10:06:20      *      *      *      *      *      BUILD: 070128W (Feb 9 2007)
03/16/2007 10:06:20      *      *      *      *      *****
03/16/2007 10:06:20
```

It is also possible to verify the Teradata ARC version number with operating-specific commands. For example, `pkginfo -x arc` on MP-RAS and `rpm -q arc` on Linux. On

Windows, click **Start>Control Panel>Add or Remove Programs** to display a list of installed programs.

Teradata ARC-Specific Terminology

The terms *backup* and *dump* are often used interchangeably with *archive*.

Restore, which is a specific Teradata ARC keyword and the name of a Teradata ARC operation, is part of the recovery process. In addition to restore, *recovery* entails other operations, such as returning data tables to their state following their modification (called rolling forward), returning data tables to the state they were in before they were modified (called rolling back), and so on. For further explanation, see [Chapter 6: “Archive/Recovery Control Language.”](#) There is no Teradata ARC statement called “recover.”

The difference between *copy* and *restore* is in the kind of operation being performed:

- A restore operation moves data from archived files back to the same Teradata Database from which it was archived or to a different Teradata Database *so long as the database DBC is already restored*.
- A copy operation moves data from an archived file back to a Teradata Database, not necessarily to the same system, *and* creates a new table if one does not already exist on the target database. When you copy selected partitions, the table must exist and be a table that was previously copied as a full-table copy.

How Teradata ARC Works

Teradata ARC creates files when you archive databases, individual data tables, selected partitions of primary partition index (PPI) tables, or permanent journal tables from the Teradata Database. You provide Teradata ARC with such files when you restore databases, individual data tables, partitions of tables, or permanent journal tables back to the Teradata Database.

Teradata ARC also includes recovery with rollback and rollforward functions for data tables defined with a journal option. Moreover, you can checkpoint these journals with a synchronization point across all AMPs, and delete selected portions of the journals.

Starting Teradata ARC

Teradata ARC runs in either online or batch mode on:

- IBM MVS
- IBM VM
- Linux RedHat
- Linux SUSE
- NCR UNIX MP-RAS
- Windows 2000/XP
- Windows Server 2003

Although Teradata ARC is normally started in batch mode, it can be run interactively. In the interactive mode, do not expect a user-friendly interface in online sessions.

Teradata ARC is started with a program module called ARCMAN.

Uses of Teradata ARC

- Archive a database, individual table, or selected partitions of a PPI table from a Teradata Database to a client resident file.
- Restore a database, individual table, or selected partitions of a PPI table back to a Teradata Database from a client resident archive file.
- Copy an archived database, table, or selected partitions of a PPI table to a Teradata Database on a different hardware platform than the one from which the database or table was archived.
- Place a checkpoint entry in a journal table.
- Recover a database to an arbitrary checkpoint by rolling it back or rolling it forward, using change images from a journal table.
- Delete change image rows from a journal table.

What is ARCMAN?

ARCMAN is the program name of the Teradata ARC utility. ARCMAN uses these files:

- *Input* (required) contains archive and recovery control statements that you create.
- *Output log* contains all runtime messages that are output from the utility. The file indicates, statement by statement, the activity that occurs from the statements in the input file. This file is generated automatically.
- *Restart log* contains restart recovery information for internal use. The utility places into this non-readable file the information that it needs if a task is terminated prematurely and then restarted (via the RESTART runtime parameter). This file is automatically generated.
- *Archive* contains archival data. An output archive file is the target of the data provided by an **ARCHIVE** statement. An input archive file is the source of data for restoring a database or table. Any number of archive files can be used during a utility run. Use the FILE parameter of the **RESTORE/COPY** or **ARCHIVE** statement to identify the names of input and output archive files. This file is required only if the task involves the creation of an archive or the restoration of a database or table.

Although Teradata ARC is able to process up to 350 ARCMAN jobs at a time, each with up to 1024 sessions, it is highly recommended that you avoid running so many jobs.

Starting ARCMAN

To view the actual JCL provided with your release, see the file on the release tape called *ARCJOB* in *dbcpfx.PROCLIB*.

Starting ARCMAN from MVS

This sample JCL shows how to start Teradata ARC from MVS:

```
//ARCJOB PROC ARCPARM=,DBCPIX=,

//USERJOB JOB <job info>
//ARCJOB PROC ARCPARM=,DBCPIX=,
//          DUMP=DUMMY,DUMPDSN=,DSEQ=,DVOL=,
//          RESTORE=DUMMY,RSTRDSN=,DBCLOG=
//*
//STEP1 EXEC PGM=ARCMAN,
//          PARM='&ARCPARM'
//STEPLIB DD DSN=&DBCPIX..AUTHLOAD,DISP=SHR
//          DD DSN=&DBCPIX..TRLOAD,DISP=SHR
//ARCHIVE DD &DUMP,DSN=&DBCPIX..&DUMPDSN,DISP=(,CATLG),
//          UNIT=TAPE,LABEL=(&DSEQ,SL),
//          DCB=BLKSIZE=32760,VOL=SER=(&DVOL)
//ARCIN DD&RESTORE,DSN=&DBCPIX..&RESTORE&RSTRDSN,DISP=OLD
//DBCLOG DD DSN=&DBCPIX..&DBCLOG,DISP=OLD
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//          PEND
//*
//S1 EXEC ARCMAN,ARCPARM='SESSIONS=100 HEX',
//          DBCPIX=DBCTEST,DUMP='DUMP.DATA',DSEQ=1,
//          DVOL='TAPE01,TAPE02',DBCLOG='ARCLOG.DATA'
//SYSIN DD DATA,DLM=##

LOGON DBC,DBC;
ARCHIVE DATA TABLES (PERSONNEL) ALL,
        RELEASE LOCK,
        INDEXES,
        FILE=ARCHIVE;

LOGOFF;
##
//
```

The JCL example uses these syntax elements:

- DUMP is the archive keyword, with DUMPDSN as the data set name for the archive output file. DUDISP specifies whether to KEEP or CATLG the output file.
- DBCLOG is the log file data set name created by Teradata ARC.
- DBCPIX is the high-level qualifier for Teradata Database load libraries.

The database DBCLOG card indicates the restart log file, and the data set must have RECFM either F, FBS or FS. For better results, set BLKSIZE near 32K. The ARCHIVE card is not needed in this example; it is shown for illustration purposes. Create and catalog database DBCLOG once and reuse it as needed. To create the database DBCLOG file, see the file on the release tape called *ARCLOG*, in *dbcpfx.PROCLIB*.

Calculating Region Size

[Table 1](#) is applicable to Teradata ARC 8.0 or earlier. Use the information in this table to estimate the minimum region size (in kilobytes) needed to run Teradata ARC from MVS.

Note: This table provides estimates for archive or restore operations only. Copy operations may require more memory.

Table 1: Minimum Region Sizes Running from MVS

Number of Sessions	Region Size For 200 Objects	Region Size For 2,000 Objects	Region Size For 8,000 Objects
10	2,500 KB	3,200 KB	5,620 KB
50	4,420 KB	5,140 KB	7,540 KB
100	6,820 KB	7,540	9,940 KB

where

Table element...	Is the...
Number of Objects	number of databases plus number of tables and stored procedures to be archived, restored, or copied. The ALL keyword expands the number according to this query: SELECT count (*) from database DBC.CHILDREN WHERE PARENT = '<db>'
Sessions	number of data sessions to use in the operation.

Example

In this example, database db2 and table db3.a are not children of database db.

```
ARCHIVE DATA TABLES (db) ALL, (db2), (db3.a),
RELEASE LOCK,
FILE=ARCHIVE;
```

If the SELECT query returns 5 for db, the total number of objects in the **ARCHIVE DATA TABLES** statement is 7.

Starting ARCMAN from VM

To start Teradata ARC from VM, include the following lines in the EXEC:

```
/* Minimal EXEC to run ARCMAN */
'GLOBAL LOADLIB LSCRTL'
'GLOBAL TXTLIB CLI'
'SET LDRTBLS 10'
'CP SET TIMER REAL'
'FILEDEF DBCLOG DISK DBCLOG FILE A (RECFM F LRECL 32760'
'FILEDEF ARCHIVE TAP1 SL (BLKSIZE 32760 RECFM U DEN 6250'
'LABELDEF ARCHIVE VOLID ARC002 VOLSEQ 0001 SEC 0 FID ARCH100'
'ARCMAN <ARC.CNTL.A SESSIONS=4'
```

The files *ARCMAN MODULE*, *CLI TXTLIB*, and *LSCRTL LOADLIB* must be on minidisks or SFS directories that are accessed from the CMS virtual machine. The latter two files are the CLIv2 and SAS/C runtime libraries, respectively. The parameter *<ARC.CNTL.A* in the

ARCMAN command indicates the utility input statements are to be read from the file *ARC CNTL* on the CMS A-disk. Output from the job is routed to the terminal by default. Route output to a file called *JOB OUTPUT* on the A-disk by adding the parameter *>JOB.OUTPUT.A* to the **ARCMAN** command.

Note that the database DBCLOG file needs to be RECFM F for VM. Set BLKSIZE near 32K. Create and catalog database DBCLOG once and reuse it as needed. The **ARCHIVE FILEDEF** and **LABELDEF** statements illustrate settings you can use if archiving to tape on VM. The ARCLLOG EXEC from the release tape allocates and initializes the archive/recovery restart log file database DBCLOG.

For the complete EXEC sample and variables, see the file on the *ARC EXEC* release tape.

Starting ARCMAN from Linux, MP-RAS, and Windows 2000/XP/Server 2003

Examples in this section are applicable to Linux, MP-RAS, and Windows 2000/XP/Server 2003. The first example is a command-line invocation of Teradata ARC from Windows 2000/XP/Server 2003:

```
ARCMAN SESSIONS=8 CATALOG OUTLOG=ARCALL.OUT <ARCALL.IN
```

The above command line calls the ARCMAN executable, uses eight sessions, and enables the catalog feature.

ARCALL.IN is an input file that contains ARCMAN commands. The “<” redirects the input file to ARCMAN.

By default, data is written to (or read from) a disk file in the current directory. With a tape product, use IOMODULE and IOPARM command-line parameters. To determine the proper values for the IOMODULE and IOPARM command-line parameters, see [“IOMODULE” on page 105](#) and [“LOGON” on page 107](#).

Two of the Open Teradata Backup (OTB) products, BakBone NetVault and Veritas NetBackup, provide additional archive and restore functionality to Teradata ARC, including a graphical user interface (GUI) to perform archive and restore operations. See the *BakBone NetVault APM/Plugin User's Guide for Teradata* or the *Veritas NetBackup Extension for Teradata Administrator Guide* for information on how to install, configure, and use the Teradata access module for these products.

Using a Configuration File or Environment Variables

It is possible to save frequently used command-line options in an environment variable or configuration file. For example:

If *T:\DEFAULTS\CONFIG.ARC* contains the following runtime parameters (the parameters are defined in this file):

```
CATALOG  
FILEDEF=(ARCHIVE,ARCHIVE_%UEN%)  
SESSIONS=8
```

and one or both of the following environment variables:

- ARCDFLT
- ARCENV or ARCVX (These are the same environment variables, except that ARCVX has override priority.)

are set *temporarily* at the command prompt as follows:

```
SET ARCDFLT=T:\DEFAULT\CONFIG.ARC
SET ARCENV=WDIR=C:\TEMP\ARC\
```

or set *permanently*, using **Start > Settings > Control Panel > System > Environment**, then ARCMAN can be called, as follows:

```
ARCMAN RLOG=JOB980813 <INPUT.ARC
```

In the above command, runtime parameters RLOG=JOB980813 are not defined in a configuration file or in an environment variable because they are job-specific. The call to ARCMAN is equivalent to the following command-line invocation of Teradata ARC:

```
ARCMAN CATALOG
FILEDEF=(ARCHIVE,ARCHIVE_%UEN%) SESSIONS=8 WDIR=C:\TEMP\ARC
RLOG=JOB980813
<INPUT.ARC
```

When called, ARCMAN automatically uses the context of any set environment variable. The example assumes that ARCENV is set to `WDIR=C:\TEMP\ARC\`.

Using Environment Variables

Almost all runtime parameters can be set as defaults in the ARCENV environment variable, or in the file pointed to by the ARCDFLT environment variable, or in some combination of the two.

These environment variables are known to ARCMAN, which loads the contents of ARCENV internally, or the Default file when building its runtime parameters. If a parameter is specified multiple times in different places, the override priority is:

- 1 Parameters set in ARCVX
- 2 Actual runtime parameters on the command line
- 3 Parameters set in ARCENV
- 4 Parameters set in the default file pointed to by ARCDFLT

For example, if you want everyone to use a specific runtime parameter, such as CATALOG, create a network default file that contains the parameter, then point ARCDFLT to it.

Local defaults or frequently used parameters such as IOMODULE can be set in ARCENV, which overrides parameters set in the network default file. Parameters on the command line override parameters set in ARCENV; parameters set in ARCVX override actual runtime parameters on the command line.

Canceling Teradata ARC

Use standard operating system or terminal commands to cancel a Teradata ARC task.

To resume an aborted operation, specify the RESTART parameter when you resubmit the task. If a restore operation aborts, there is no need to delete the partially restored database because partially restored data is overwritten when you retry the restore operation.

Because utility locks are not released when an operation is aborted, you must explicitly release the locks on the entity that was being processed when the operation aborted.

Sample Teradata ARC Scripts

Example 1: Archive Script

```
LOGON DBCID/USERID,PASSWORD;  
ARCHIVE DATA TABLES (DB1) ALL,(DB2),  
RELEASE LOCK,  
FILE=ARCHIVE;  
LOGOFF;
```

This script performs these archive operations:

- 1 ARCMAN logs on to a Teradata Database pointed to by database DBCID.
- 2 ARCMAN places Host Utility (HUT) locks on the databases to be archived before starting the archive operation.
- 3 The database's DB1, all its descendent databases, and DB2 are backed up. The operation is performed database by database in alphabetical order.
- 4 The result of the backup is written to the file pointed to by FILE=.

For Linux, MP-RAS, and Windows 2000/XP/Server 2003: If no IOMODULE and IOPARM runtime parameters are specified (as in the above example), data is written to hard disk.

If IOMODULE and IOPARM are specified, data is written to the tape drive or other media specified and pointed to by IOMODULE and defined by the initialization string to which IOPARM is set.

For an example, see [“Starting ARCMAN from Linux, MP-RAS, and Windows 2000/XP/Server 2003” on page 20](#).

- 5 The RELEASE LOCK option releases the HUT locks after the archive of each database is complete.
- 6 After all databases are archived, the LOGOFF command disconnects ARCMAN from the Teradata Database.

Example 2: Restore Script

```

LOGON DBCID/USERID,PASSWORD;
RESTORE DATA TABLES (DB1) ALL, (DB2),
RELEASE LOCK,
FILE=ARCHIVE;
LOGOFF;

```

This script performs these restore operations:

- 1 ARCMAN logs on to a Teradata Database pointed to by database DBCID.
- 2 ARCMAN places Host Utility (HUT) locks on the databases to be restored before starting the restore operation.
- 3 The database's DB1, all its descendent databases, and DB2 are restored. The operation is performed database by database in alphabetical order.
Note: For each database, these operations are performed table by table in alphabetical order:
 - Primary data is restored.
 - If applicable, fallback and index data are built from primary data.
- 4 The RELEASE LOCK option releases the HUT locks after the restore of each database is complete.
- 5 After all databases are restored, the **LOGOFF** command disconnects ARCMAN from the Teradata Database.

Data Extension Modules

Note: Data extension modules are valid only on Windows platforms.

Data extension modules expand the functionality of Teradata ARC by allowing archive data to be processed (encrypted, for example) prior to being sent to, or after being retrieved from, output media.

The **ENABLE DATA EXTENSION** command dynamically loads extension modules; the **ENABLE ENCRYPTION** command dynamically loads the encryption extension module. The Protegrity® encryption module, *pepbar.plm*, is the default encryption module. Refer to Protegrity documentation at <http://www.protegrity.com/solutioncenter.html> for information on installation, setup, and operation of the Protegrity product.

It is also possible to enable data extension modules with command-line parameters DEMODULE and DEPARM. When using the parameters, specify both DEMODULE and DEPARM (which correspond with the MODULE and PARM options in the **ENABLE DATA EXTENSION** command).

Example

```

ARCMAN SESSIONS=8 CATALOG OUTLOG=ARCALL.OUT DEMODULE=PEPBAR.PLM
DEPARM='ALGORITHM=AES128'

```

The example has the same result as specifying:

```
ENABLE ENCRYPTION;
```

In the **ENABLE ENCRYPTION** example, the *pepbar.plm* Protegrity encryption module and AES128 encryption algorithm are used because they are the defaults.

Archive/Recovery Operations

This chapter contains these topics:

- [Database DBC](#)
- [Database SYSUDTLIB](#)
- [Archiving Tables and Databases](#)
- [Restoring Tables and Databases](#)
- [Recovering Tables and Databases](#)
- [Copying Tables and Databases](#)
- [Using Host Utility Locks](#)
- [Setting Up Journal Tables](#)
- [Controlling Journal Checkpoint Operations](#)

Database DBC

The DBC database contains critical system tables that define the user databases in the Teradata Database.

[Table 2](#) lists the tables that are archived for database DBC.

Note: If a DBC user table is not listed in [Table 2](#), it will not be archived for database DBC. Instead, use SQL to copy the DBC user table into a user-level table in another database for Teradata ARC to back up. The information cannot be copied back to the DBC user table during a restore process.

Table 2: Tables Archived for Database DBC

Table Name	Description
AccessRights	Specifies all granted rights.
AccLogRuleTbl	Defines access logging rules generated by executing BEGIN/END LOGGING statements.
Accounts	Lists all authorized account numbers.
CollationTbl	Defines MULTINATIONAL collation.
DBase	Defines each database and user.

Table 2: Tables Archived for Database DBC (continued)

Table Name	Description
Hosts	Defines information about user defined character sets used as defaults for client systems.
LogonRuleTbl	Defines information about logon rules generated by a GRANT LOGON statement.
Next	Generates table, stored procedure, and database identifiers (internal table).
OldPasswords	Lists passwords that are no longer in use, including the user to which the password was assigned, the date the password was changed, and encrypted password string.
Owners	Defines all databases owned by another.
Parents	Defines all parent/child relationships between databases.
Profiles	Defines the available roles and profiles on the Teradata machine.
RCConfiguration	Records the configuration for RCEvent rows.
RCEvent	Records all archive and recovery activities.
RCMedia	Records all removable devices used in archive activities.
RepGroup	Defines each replication group in the server.
RoleGrants	Defines the available roles and profiles on the Teradata machine.
Roles	Defines the available roles and profiles on the Teradata machine.
SysSecDefaults	Defines the system security defaults of a Teradata Database, for example, the minimum and maximum characters allowed in a password.
Translation	Defines hexadecimal codes that form translation tables for non-English character sets.
UDTCast	Contains information on the source and target data types that are involved in the casting operation.
UDTInfo	Captures the specifics contained within the CREATE TYPE statement.
UDTTransform	Contains the transform group name and the routine identifiers.

If included in an archive operation, database DBC is always archived first, regardless of alphabetical order. If included in a restore or copy operation, database DBC is always restored or copied first. Refer to the next section for specific instructions on restoring database DBC.

Restoring

To restore database DBC to a system, the DBC database must be the only database that contains objects. To remove all user tables, views, macros, stored procedures, triggers, UDFs, and UDTs, use this command:

```
delete database (DBC) ALL, exclude (DBC);
```

To drop permanent journal tables, use either the **MODIFY USER** or **MODIFY DATABASE** statement (See the MODIFY DATABASE/USER description in *SQL Reference: Data Definition Statements*.)

Database SYSUDTLIB

There is a system database, SYSUDTLIB, that is created and maintained by the Teradata Database and contains the definition of all of the User Defined Types (UDTs) defined in the Teradata Database. Database SYSUDTLIB is logically linked to the DBC database. Therefore, whenever an operation involves the DBC database, the SYSUDTLIB database is *usually* involved also. The next paragraphs discuss exceptions.

Archiving

If you archive database DBC, ARC automatically archives database SYSUDTLIB also. DBC is always archived first and SYSUDTLIB archived second, before any other objects are archived. Do not archive SYSUDTLIB as a separate object.

If you use the EXCLUDE option to exclude DBC on an **ARCHIVE** statement, SYSUDTLIB will be excluded also.

Restoring

If you restore database DBC, ARC automatically restores database SYSUDTLIB too. DBC is always restored first and SYSUDTLIB restored second, before any other objects are restored. Do not restore SYSUDTLIB as a separate object.

If you use the EXCLUDE option to exclude DBC on a **RESTORE** statement, SYSUDTLIB will be excluded also.

Refer to [“Restoring” on page 27](#) for specific instructions on restoring database DBC.

Copying

You cannot specify database DBC or SYSUDTLIB on a **COPY** statement unless you use the **COPY FROM** format. When using the **COPY FROM** format, copy either DBC or SYSUDTLIB as separate source objects, but only if the target database name is not DBC or SYSUDTLIB.

Deleting

To prepare a system for restoring Database DBC, use the **DELETE** command to delete all objects in the system except for permanent journal tables. (To drop permanent journal tables, use either the **MODIFY USER** or **MODIFY DATABASE** statement.)

```
delete database (DBC) ALL, exclude (DBC);
```

When the EXCLUDE DBC option is used with the (DBC) ALL option on a **DELETE DATABASE** statement, SYSUDTLIB will be delinked from DBC. This allows SYSUDTLIB to be deleted so that DBC can be restored. However, SYSUDTLIB is deleted last, after all other databases have been deleted. Any objects that have definitions based on the UDTs stored in SYSUDTLIB are deleted before the UDTs themselves are deleted.

If you use the EXCLUDE option to exclude DBC on a **DELETE DATABASE** statement, but do not use the (DBC) ALL option, SYSUDTLIB will continue to be linked to DBC and both DBC and SYSUDTLIB will be excluded from the delete operation.

Do not specify SYSUDTLIB as an object of the **DELETE DATABASE** statement.

Archiving Tables and Databases

An archive operation copies database information from the Teradata Database to one or two client resident files. The archive can be from any of the following:

- All AMPs
- Specified clusters of AMPs
- Specified AMPs

In general, Teradata ARC archives have these characteristics:

- If an **ARCHIVE** statement specifies more than one database, Teradata ARC archives the databases (except for DBC and SYSUDTLIB) in alphabetical order. Within each database, all related tables and stored procedures are archived in alphabetical order.
- If the statement specifies multiple individual tables, Teradata ARC archives the tables in alphabetical order first according to database and then by table.
- If an archive of database DBC is interrupted for any reason, the entire archive must be performed again. It is not possible to restart an archive operation on database DBC.
- Archives cannot contain both data tables and journal tables. A journal table archive contains only the saved portion of the table.
- Journal table archives do not delete change images from the journal table. Therefore, you can accumulate change images for several activity periods by executing a checkpoint with save after each period and then archiving the table without deleting the journal.

Concurrency Control

An **ARCHIVE** statement places a HUT read lock on:

- the database when archiving the database
- the table when archiving the table

If there is an existing SQL read lock or access lock (created by using the **LOCKING FOR ACCESS** modifier) on the database, it is still possible for Teradata ARC to obtain its read lock and start the archive.

Introduction to Teradata Warehouse contains details on concurrency control and transaction recovery.

Archiving With All AMPs Online

An all-AMPs archive (or dictionary archive) of a database or table contains the Teradata Database dictionary rows that are needed to define the entity.

[Table 3](#) alphabetically lists the dictionary rows that are archived by an all-AMPs data archive (or a dictionary tables archive) of a user database or table.

Table 3: Dictionary Rows Archived for User Database

Table Rows	Description
ConstraintNames	Constraints that have names
Dependency	Stores relationships among a UDT, its dependent routines, User-Defined Casts, User-Defined Transforms, User-Defined Orderings, and any dependency on any other database object.
Indexes	Columns that are indexes
IndexNames	Indexes that have names
ReferencedTbls	Referenced columns (i.e., Parent Key) for Parent tables
ReferencingTbls	Referencing columns (i.e., Foreign Key) for Child tables
TableConstraints	Table level Check constraints
TVFields	All columns in data tables and views
TVM	All data tables, views, macros, stored procedures, and triggers in the database
UnresolvedReferences	Unresolved referential constraints

Example

The following example shows how to archive all databases when all AMPs are online:

```
LOGON DBC, DBC;  
ARCHIVE DATA TABLES (DBC) ALL,  
RELEASE LOCK,  
FILE=ARCHIVE;  
LOGOFF;
```

When the archive operation is complete, check the output log to verify that all AMPs remained online during the archive.

If an AMP goes offline during an archive, the output log reports the processor number that is offline. If one or more AMPs go offline, see [“Archiving With Offline AMPs” on page 36](#) for more information.

Archiving Selected Partitions of PPI Tables

You can perform an all-AMPs archive on one or more partitions of a table rather than performing a full-table backup and restore. The ability to select partitions from PPI tables is limited to all-AMP archives. Dictionary, cluster, and journal archives are not supported.

Use partitioning to:

- Archive only a subset of data and avoid archiving data that has already been backed up. (This can minimize the size of the archive and improve performance.)
- Restore data in a table that is partially damaged.
- Copy a limited set of data to a disaster recovery machine or to a test system.

Before using this feature, be sure to understand [“Potential Data Risks When Archiving/Restoring Selected Partitions” on page 33](#).

For information about the keywords that are specific to archiving and restoring selected partitions of PPI tables, see [“Using Keywords with ARCHIVE” on page 142](#) and [“Archiving Selected Partitions of PPI Tables” on page 146](#).

Description

PPI allows the division of data in a table into separate partitions based on a specific partitioning scheme. With Teradata ARC, individual partitions can be archived according to user-defined partitioning expressions. For more information about options for PPI archives/restores, see [Chapter 6: “Archive/Recovery Control Language”](#).

Considerations

Consider the following when archiving selected partitions in PPI tables:

- A restore operation always deletes the selected partitions of the target table before restoring the rows that are stored in the archive.
- Archiving selected partitions operates on complete partitions within tables, meaning that the selection of a partial partition implies the entire partition.
- PPI and non-PPI tables are allowed in a single command. This allows you to manage both table types in a single database with the EXCLUDE TABLES option.
- Partitioning is based on one or more columns specified in the table definition.

- Partition elimination restricts a query to operating only in the set of partitions that are required for the query.
- Incremental archives are possible by using a partition expression that is based on date fields, which indicate when a row is inserted or updated.
- An archive or restore of selected partitions only places full-table HUT locks. HUT locks on individual partitions are not supported.
- It is recommended that you re-collect table statistics after a restore of selected partitions because statistics are part of the table dictionary rows, which are not restored during a partition-level restore.
- If a table has a partitioning expression that is different from the partitioning expression used in the PPI archive, a PPI restore is possible as long as no other significant DDL changes are made to the table.

The archival of selected partitions has limitations. For more information, see [“Potential Data Risks When Archiving/Restoring Selected Partitions” on page 33](#) and [“Considerations Before Restoring Data” on page 40](#).

The next example shows a partitioning expression that follows the PARTITION BY keyword. Data is partitioned for the TransactionHistory table, based on the month when the transaction occurred:

```
CREATE TABLE TransactionHistory
(TransactionID          INTEGER,
 TransactionDate        DATE FORMAT 'yyyy-mm-dd',
 TransactionParam1      INTEGER,
 ...
)
PRIMARY INDEX (TransactionID)
PARTITION BY RANGE_N
(TransactionDate BETWEEN DATE '2000-01-01' AND DATE '2004-12-31'
EACH INTERVAL '1' MONTH
);
```

Procedure for Backing Up Selected Partitions

The procedure in this section is a generic example of how to set up archive and restore scripts for selected partitions. This example is based on the TransactionHistory table previously described.

Assume that this backup schedule is desired for the TransactionHistory table:

- An incremental backup of the currently active partition will be done nightly.
- At the beginning of each month, the final incremental backup for the previous month will be done; this backup will be saved until the next differential or full-table backup is done.
- Every three months, a differential backup will be done, containing the data for the last three months.
- Every year, a full-table backup will be done.

To back up PPI data in the TransactionHistory table:

1 Perform a full-table archive:

```
ARCHIVE DATA TABLES
  (SYSDBA.TransactionHistory),
RELEASE LOCK,
FILE=ARCHIVE;
```

2 Set up incremental archives:

```
ARCHIVE DATA TABLES
  (SYSDBA.TransactionHistory)
  ( PARTITIONS WHERE
    (! TransactionDate BETWEEN CURRENT_DATE - 3 AND CURRENT_DATE
      ! ) ),
RELEASE LOCK,
FILE=ARCHIVE;
```

Note: In this example, 'CURRENT_DATE - 3' archives a partition even after it becomes non-active, in case the final archive of the partition fails or the value of CURRENT_DATE changes during the final backup.

3 Set up differential backups:

```
ARCHIVE DATA TABLES
  (SYSDBA.TransactionHistory)
  ( PARTITIONS WHERE
    (! TransactionDate BETWEEN DATE '2004-01-01' AND DATE '2004-03-31'
      ! ) ),
RELEASE LOCK,
FILE=ARCHIVE;
```

4 (Optional) Perform a separate partition backup if you update a partition that is not archived by the incremental backup step (step 2):

```
ARCHIVE DATA TABLES
  (SYSDBA.TransactionHistory)
  ( PARTITIONS WHERE
    (! TransactionDate = DATE '2004-03-15'
      ! ) ),
RELEASE LOCK,
FILE=ARCHIVE;
```

To fully restore the TransactionHistory table:

1 Perform a complete restoration of the full-table archive:

```
RESTORE DATA TABLES
  (SYSDBA.TransactionHistory),
RELEASE LOCK,
FILE=ARCHIVE;
```

2 Perform a full restoration of the differential and incremental archives (in order):

```
RESTORE DATA TABLES (SYSDBA.TransactionHistory) (ALL PARTITIONS),
RELEASE LOCK,
FILE=ARCHIVE;
```


- 3 (Optional) If a separate partition backup is performed due to an update of a non-active partition, restore the separate partition backup after (or instead of) the differential or incremental backup that contains the updated partition:

```
RESTORE DATA TABLES
(SYSDBA.TransactionHistory)
( PARTITIONS WHERE
(! TransactionDate = DATE '2004-03-15'
!) ),
RELEASE LOCK,
FILE=ARCHIVE;
```

Individual partitions can also be restored from a full-table or selected-partition archive. To restore individual partitions, specify the partitions to be restored in a PARTITIONS WHERE expression:

```
RESTORE DATA TABLES
(SYSDBA.TransactionHistory)
( PARTITIONS WHERE
(! TransactionDate BETWEEN DATE '2004-05-01' AND DATE '2004-05-31'!)
),
RELEASE LOCK,
FILE=ARCHIVE;
```

Potential Data Risks When Archiving/Restoring Selected Partitions

Be careful when archiving partitioned tables: a number of undesirable conditions can occur. For additional issues that might occur during restore operations, see [“Considerations Before Restoring Data” on page 40](#).

Caution: The following cases generally do not display an error or give any indication that a problem has occurred. In most instances, the only indication is that data is incorrect or is missing from a table.

- **Use Correct Specifications** - The incorrect use of specifications causes the following problems:
 - An incorrect PARTITIONS WHERE specification during backup can result in an incomplete archive or difficulties during a restore operation.
 - An incorrect PARTITIONS WHERE or ALL PARTITIONS specification during restore can result in data lost from a table or the restoration of stale data to a table if the archive being restored contains partial, incomplete, or stale versions of an already existing partition.
- **Restrict Updates to Active Partitions** - It is not possible to determine which partitions have been modified since the last backup. If changed partitions are not re-archived, the changes are lost when restored.

For example, if, for a given table, the backup strategy is to only backup the active (latest) partition of the table, and a change is made to a non-active partition (to fix an incorrect update), the change is not archived unless you run a separate archive of the changed partitions.

The remedy for this situation is either to restrict updates to the active partitions only (by using views to control which rows/partitions are updated) or to re-archive all modified partitions.

- **Do Not Change Values of Functions or Variables** - If a built-in SQL function or variable is used in the PARTITIONS WHERE condition, and the value of the function or variable changes during the job, a different set of partitions might be archived (or restored) for some objects in that single archive.

For example, if an archive job uses the CURRENT_DATE built-in function to determine which is the active partition, and the backup runs past midnight, the date change causes a different partition to be selected. This means that objects archived after midnight will archive the new (and probably empty) partition.

The remedy for this situation is to do one of the following:

- Avoid using a changing function or variable in the PARTITIONS WHERE condition.
- Run the backup at a time when the value will not change.
- Modify the PARTITIONS WHERE condition to take the value change into account when selecting partitions. For example, you can define a range, such as `'BETWEEN CURRENT_DATE - n AND CURRENT_DATE'` to archive the active partition even if the date changes.

- **Always Specify PARTITIONS WHERE or ALL PARTITIONS** - If PARTITIONS WHERE or ALL PARTITIONS are not specified for a RESTORE or COPY operation, the default action is to overwrite the entire table with the archived table definition and data.

Essentially, this is the same as a full-table restore.

For example, if you forget to use PARTITIONS WHERE when you try to restore a single-partition backup, data is dropped from the table and the single partition stored on the archive is restored.

The remedy for this situation is to always specify PARTITIONS WHERE or ALL PARTITIONS when restoring partitions into an existing table, unless you intended to overwrite the existing table.

- **Know What Partitions are Being Deleted** - In a RESTORE or COPY operation, all partitions that match the PARTITIONS WHERE condition are deleted, even if they are not stored on the archive.

For example, if you restore an archive that contains the data for April 2004, but mistakenly enter a PARTITIONS WHERE condition that matches both March and April 2004, the data for both March and April 2004 are deleted, and only April 2004 is restored.

The remedy for this situation is to be very careful about using the PARTITIONS WHERE condition. If there is any doubt about which partitions are affected, COPY the selected partition backup to a staging table, and manually copy the desired partition(s) into the target table using INSERT/SELECT and/or DELETE.

- **Avoid Restoring From a Previous Partitioning Scheme** - When changing the partitioning expression for a table, it is possible to change the boundaries of existing partitions. If these partitions are restored, Teradata might either drop more data than expected or restore less data than expected, if the archive does not include data for all of the selected partitions.

For example, if an archive is done on a table partitioned by month with the archive data corresponding to March 2004, and the table is re-partitioned by week, then a PPI restore of the March backup (using ALL PARTITIONS) overwrites the data for all weeks that contain

at least one day in March. As a result, the last few days of February and the first few days of April might be deleted and not restored.

The remedy for this situation is to avoid restoring partition backups from a previous partitioning scheme to an updated table. Or, use LOG WHERE for the weeks that contain days in both March and February/April, and manually copy the rows into the table.

- **Track the Partitions in Each Archive** - Manual steps are required to determine which partitions are archived by a given backup job, or to determine which backup job has the latest version of a given partition. ANALYZE displays the Teradata-generated bounding condition that defines the archived partitions. (This differs from a user-entered condition that might only qualify partial partitions.) In this case, inconsistent or old data might be restored to the table if the wrong archive is restored for a partition, or if partition-level archives are restored out-of-order and the archives contain an overlapping set of partitions.

For example, updated data is lost in the following situation. Assume that a final backup for a March 2004 partition is performed on April 1, 2004. On April 5, a mistake is found in a row dated March 16, so the row is updated, and a new backup of the March partition is done. If, for instance, the table is accidentally deleted a month later, and you attempt to restore the April 1 backup, instead of the April 5 backup, the updated data is lost.

The remedy is to either keep track of the partition contents of each archived table, retain the output listing associated with a tape, or run ANALYZE jobs on archives to determine the partitions in each archive.

Archiving by Cluster

An alternative to archiving data tables from all AMPs into a single archive is to archive into a set of archive files called a *cluster archive*.

To create a cluster archive, archive the data tables by groups of AMP clusters so that the complete set of archive files contains all the data from all of the AMPs. This allows you to save time by archiving jobs in parallel.

Cluster Archive vs. Dictionary Archive

A cluster archive does not contain dictionary information (the data stored in the dictionary tables for the tables in the database). Create a dictionary archive before you run a cluster archive for the first time and whenever you modify the structure of tables in the cluster archive. Consequently, if you archive by cluster, archive the dictionary and maintain it separately. Do not create a cluster archive of journal tables or database DBC.

Because cluster archives only contain table data, perform a dictionary archive for any databases or tables that are included in the cluster archive:

- 1 Create the dictionary archive immediately prior to the cluster archives and then, without releasing Teradata ARC locks.
- 2 Follow with a set of cluster archive requests covering all AMPs.

To restore successfully, the table data in the cluster archives must match the dictionary definition contained in the dictionary archive. A restore operation fails when it tries to restore cluster archives that do not match the dictionary archive.

Example

This example assumes the system has four clusters, each consisting of two AMPs. Each archive is of two clusters. The procedure is divided into three jobs. Run Job 1 first, then run Job 2 and Job 3 in parallel. Finally, run Job 4. This example makes a dictionary archive and two cluster archives.

Job 1	<pre>LOGON USER,USER; ARCHIVE DICTIONARY TABLES (USERDB), FILE = ARCHDICT; LOGOFF;</pre>
Job 2 (run in parallel with Job 3)	<pre>LOGON USER, USER; ARCHIVE DATA TABLES (USERDB), CLUSTER = 0,1, FILE = CLUSTER1; LOGOFF;</pre>
Job 3 (run in parallel with Job 2)	<pre>LOGON USER, USER; ARCHIVE DATA TABLES (USERDB), CLUSTER = 2,3, FILE = CLUSTER2; LOGOFF;</pre>
Job 4	<pre>LOGON USER, USER; RELEASE LOCK (USERDB); LOGOFF;</pre>

Archiving With Offline AMPs

The Teradata Database configuration determines how to archive when AMPs are offline. If a table has fallback protection or dual after-image journaling, Teradata ARC makes a complete copy of the data rows, even if an AMP (per cluster) is offline during the archive. For a cluster archive, only AMPs within specified clusters are considered.

Because the Teradata Database dictionary rows are all defined with fallback, Teradata ARC always performs a complete archive of the necessary dictionary entries, even if AMPs are offline. This is also true when database DBC is archived.

If AMPs are offline and you request an all-AMPs or a cluster archive involving nonfallback tables or a journal table archive that contains single images, then you must do a specific AMP archive for the offline processors after the AMPs are back online.

Because single after-images (remote or local) are maintained on single AMPs, after-images for offline AMPs are not included in the archive. If you restore one of these archives with a rollforward operation, data from some of the online AMPs is not rolled forward.

Example 1

This example archives all databases when one AMP is offline with Teradata Database. In this case, the offline AMP is *not* out of service because of a disk failure.

```
LOGON DBC, DBC;
ARCHIVE DATA TABLES (DBC) ALL,
      RELEASE LOCK,
      INDEXES,
      FILE=ARCHIVE;
LOGOFF;
```

When the archive operation is complete, check the print file to verify that the offline AMP did not come back online during the archive. If the offline AMP came back online during the archive, the print file reports the following with Teradata Database, where *n* is the number of the AMP:

```
AMP n BACK ON-LINE
```

When the offline AMP returns to online operation, run another archive procedure to archive rows from the offline AMP, even if the AMP came back online during the archive.

Example 2

This example archives all nonfallback data from an AMP that was offline during a previous Teradata Database archive operation.

```
LOGON DBC, DBC;
ARCHIVE NO FALLBACK TABLES (DBC) ALL,
      AMP=3,
      EXCLUDE (DBC) ,
      RELEASE LOCK,
      FILE=ARCHIVE;
LOGOFF;
```

Archiving Large Objects (LOBs)

Teradata ARC supports the archive operation for tables that contain large object columns as long as the database systems are enabled for large object support. However, large object columns cannot be restored on a system that uses a hash function that is different than the one used for the archive.

An archive of selected partitions with LOBs is supported, but the restore is not. To restore selected partitions of LOBs, perform a full-table restore.

Archiving Non-Hashed and Partially Loaded Tables

Each table has information about itself stored in a special row called a table header. Table headers are duplicated on all AMPs.

An archive operation captures only table headers for:

- Nonhashed tables
- Tables with loading in progress by the FastLoad or MultiLoad utilities. When these tables are restored, Teradata ARC restores them empty.

Encrypting Archived Data

To provide enhanced security, database data can be encrypted prior to being stored in an archive file. To enable this feature, specify a data encryption extension module prior to using **ARCHIVE**.

There are two ways to request data encryption:

- Specify the DEMODULE and DEPARM runtime parameters on the command line when executing an archive job.
- Specify **ENABLE DATA EXTENSION** or **ENABLE ENCRYPTION** in the archive script.

Archiving Online

Online archiving allows concurrent updates to tables during the archive process. Any updates during the archive are logged so that they can be rolled back to a consistent point during a restore.

There are different ways to start and stop online archiving. Use the **ONLINE** keyword in the **ARCHIVE** statement or the **LOGGING ONLINE ARCHIVE ON** and **LOGGING ONLINE ARCHIVE OFF** statements.

The **ONLINE** keyword in the **ARCHIVE** statement specifies objects to be archived online. Then, Teradata ARC automatically enables and disables logging as necessary. This method can be used for all-AMP archives.

The **LOGGING ONLINE ARCHIVE ON** statement explicitly starts online logging for one or more objects in the system. Subsequent **ARCHIVE** statements on these objects will perform an online archive, allowing concurrent updates. Use this method for cluster archives.

The **LOGGING ONLINE ARCHIVE OFF** statement stops online logging on the specified objects. This statement must be submitted for all objects after the online archive is complete if the **LOGGING ONLINE ARCHIVE ON** statement was used, or if an online archive job fails. If **LOGGING ONLINE ARCHIVE OFF** is not submitted, logging will continue on the objects indefinitely.

Determining Whether Online Archive Logging is in Use

To determine which tables and databases are currently being logged for online archive, run a query against the ArchiveLoggingObjsV view. An example of a query is:

```
select CreateTimeStamp,
       DatabaseName (VARCHAR(30)),
       TVMName (VARCHAR(30))
from DBC.ArchiveLoggingObjsV;
```

*** Query completed. One row found. 3 columns returned.
*** Total elapsed time was 1 second.

CreateTimeStamp	DatabaseName	TVMName
2007-04-27 16:57:37	jmd	t1

Restoring Tables and Databases

A restore operation transfers database information from archive files on the client to one of the following:

- Database DBC
- Database SYSUDTLIB
- User Database or Table
- All AMPs
- Clusters of AMPs
- Specified AMPs

Because an archive of database DBC contains the definition of all user databases in the Teradata Database, restoring database DBC to the Teradata Database automatically defines for that system the user databases from a database DBC archive.

Restoring database DBC also restores database SYSUDTLIB automatically. SYSUDTLIB contains the definition of all UDTs defined in the system.

These types of information are restored:

- An all-AMPs (dictionary) data tables archive of a user database contains all table, view, macro, stored procedure, and trigger definitions in the database.
- An all-AMPS restore of a database archive automatically restores all data tables, views, macros, stored procedures, and triggers from the archive. Similarly, a restore of a dictionary archive restores the definitions of all data tables, views, macros and triggers, and the dictionary entries for stored procedures. But it does not restore any data.
- An all-AMPS restore of selected partitions restores only the partitions specified by the PARTITIONS WHERE condition. Dictionary, cluster, and journal restores are not supported.
- Restores of selected partitions are always to existing tables, and certain changes to the target table definition are allowed. For more information, see [“Changes Allowed to a PPI Table” on page 147](#).
- Secondary indexes are also updated for restores.
- UDFs
- UDTs

Conditions Needed for a Restore Operation

- **Data Dictionary Definitions** - Archived table and databases can only be restored to a Teradata Database if the data dictionary contains a definition of the entity to be restored.
- **Locks** - Because Teradata ARC uses a utility (HUT) lock, a single user cannot initiate concurrent restore tasks. The Teradata ARC lock avoids possible conflicts when releasing locks. To restore concurrently, supply a separate user identifier for each restore task.
For restores of selected partitions, Teradata ARC applies a HUT write lock.

- **UtilVersion Match** - To restore selected partitions, the archive *UtilVersion* must match the *UtilVersion* on the target system if they are both non-zero. If *UtilVersion* is zero on the archive, the *Version* on the archive must match the *UtilVersion* on the target system. For more information, see [“Archiving Selected Partitions of PPI Tables” on page 30](#).

Considerations Before Restoring Data

Before performing a restore operation, consider the following items. For a list of potential issues regarding the archiving of selected partitions, see [“Potential Data Risks When Archiving/Restoring Selected Partitions” on page 33](#).

- **Dropped Database and Users** - A restore of a database DBC drops all new databases or users created since the time the archive was created. Additionally, all new UDTs are dropped (because database SYSUDTLIB will also be restored with database DBC).
- **Dropped Tables, Views, and Macros** - A restore of a user database drops any new tables, views, macros, stored procedures, triggers, or UDFs created since the archive of the database.
- **Restoring Undefined Tables with COPY** - Because of potentially conflicting database and table internal identifiers, you cannot restore a database or table to another system that does not contain an equivalent definition of the entity (for example, the same name and internal identifier). To restore data tables that are not already defined in the data dictionary, use the **COPY** statement.
- **Insufficient Memory for Large Tables** - Teradata ARC uses the same methodology as the Teradata SQL CREATE INDEX function to rebuild secondary table indexes. If there is insufficient available disk space, it may not be possible to restore a very large table because of the amount of temporary disk space that is required to recreate a secondary index.
- **Join Indexes** - Teradata ARC does not archive or restore join indexes. If a database containing a join index is restored, then the join index will no longer exist when the restore operation is complete. If a partial database restore is done where a table is restored, any join indexes that reference that table will be marked as invalid.

A warning occurs when a SHOW JOIN INDEX request is performed on an invalidated join index. A HELP DATABASE request lists both valid and invalid join indexes, but it will not indicate which join indexes are invalid. To re-create the join index:

- a Extract the join index definition from the SHOW JOIN INDEX output.
 - b Drop and re-create the join index.
 - c Collect new statistics.
- **Matching Hash Functions for Large Objects** - Teradata ARC supports the restore operation for tables that contain large object columns as long as the database system is enabled for large object support. However, large object columns cannot be restored on a system that uses a hash function that is different than the one used for the archive.
 - **Certain Statements Force the Restoration of the Entire Database** - A Teradata SQL **DROP** or **RENAME** statement cause the definition of an entity to be removed from the dictionary, and this same definition cannot be re-created using a Teradata SQL **CREATE** statement because a create operation is a new logical definition.

As a result, you cannot restore a dropped table unless you restore the entire database. Furthermore, you cannot restore a dropped database unless you restore database DBC first. A Database DBC archive must also have a definition of the dropped database in it. Consequently, it is recommended that you archive individual tables with care, and that you periodically archive the dictionary as well.

If you need to restore all of a user database or database DBC (that is, all of the Teradata Database) because of a catastrophic event, you can restore the dictionary information for the database at the database level before you restore the individual tables. Restoring the dictionary first restores the table definitions, so you are able to successfully restore the tables.

Restoring the Database DBC

Before restoring database DBC:

- 1 Reconfigure the system from a single AMP to the desired configuration.
- 2 Run the DBC Initialization Procedure (DIP) to initialize system views, macros, users, and error messages tables. (See the software release cover letter for information on running DIP.)
- 3 If user tables, views, macros, stored procedures, triggers, or UDFs are defined in the Teradata Database, drop them before restoring database DBC. Use the following statement to drop all such objects, except for permanent journal tables:

```
DELETE DATABASE (DBC) ALL, EXCLUDE (DBC);
```

This **DELETE DATABASE** statement also deletes all UDTs from the SYSUDTLIB database.

To drop permanent journal tables, use either the **MODIFY USER** or **MODIFY DATABASE** statement. (See the **MODIFY DATABASE/USER** descriptions in *SQL Reference: Data Definition Statements*.)

Interrupted or Failed Restores

If a restore of database DBC or SYSUDTLIB is interrupted for any reason, the entire restore must be performed again. You cannot restart a restore operation on database DBC or SYSUDTLIB. If the restore of database DBC or SYSUDTLIB is interrupted, reinitialization of the database and disks with the database software and dictionary tables might be involved (in some cases) prior to rerunning the restore of database DBC or SYSUDTLIB.

Migration Scripts

If you are migrating across a major Teradata release, it might be necessary to run a migration script after restoring database DBC. To learn how a typical archive script performs its operation, see the sample scripts in the Teradata Database migration documentation for your database version.

If a migration script is needed, Teradata ARC prints a message that specifies a script that needs to be run, and automatically exits. If this occurs, immediately start the specified script before continuing with the restore of the Teradata system. If the migration script fails or is interrupted, re-initialize the system, and restart the migration from the beginning.

System Upgrades

If you are upgrading a system, you might need to run conversion utilities. See the *Teradata DBS for UNIX Field Support Guide* for more information.

Restoring a User Database or Table

All-AMPs Restore of a Complete Database

An all-AMPs restore of a complete database from an archive of a complete database:

- 1 Drops all tables, views, macros, stored procedures, triggers, join indexes, hash indexes, and user-defined functions in the Database resident version of the database,
- 2 Restores all tables, views, macros, stored procedures, triggers, and user-defined functions in the archive.

All-AMPs Restore of a Selected Table

An all-AMPs restore of a selected data table from an archive of a complete database restores only the dictionary rows and data for the requested table. Other data tables, views, macros, stored procedures, triggers, join index, hash indexes, and user-defined functions in the database are not restored.

All-AMPs Restore of a Full Database from a Selected Table Archive

An all-AMPs restore of a full database from an archive that was created for specific data tables restores only the dictionary rows and data for the tables that were originally archived.

Restoring Selected Partitions

Selected partitions can be directly backed up and restored with a `PARTITIONS WHERE` option that restricts the list of rows processed. The `PARTITIONS WHERE` option operates on complete table partitions. A **RESTORE** (or **COPY**) operation wipes out selected partitions (specified by the `PARTITIONS WHERE` option) of an existing target table before recovering the rows stored on the backup tape. For the most part, a backup operation on selected partitions is allowed with the same all-AMP BAR styles as for full tables.

Restoring selected partitions is affected by the maintenance activities that can occur on a table. For example, a user can add or delete a column, add or delete a secondary index, or change the partitioning expression.

If a selected partition was archived prior to one of these changes, you may or may not be able to restore that partition back to the table. You are allowed to restore selected partition data back to a table even if the target table has some characteristics that are different from the source stored on tape. However, not all changes are allowed and some will prevent the restoration of the selected partition data. For a list of acceptable and non-acceptable differences, see [“Changes Allowed to a PPI Table” on page 147](#) and [“Restrictions on Archiving Selected Partitions” on page 146](#).

For an overview of backing up partitioned data with Teradata ARC, see [“Archiving Selected Partitions of PPI Tables” on page 30](#).

For a complete discussion of the keywords for restoring partitioned data, see [“Archiving Selected Partitions of PPI Tables” on page 146](#).

All-Amps Restore of Selected Partitions

The following options restore partitioned data:

- **PARTITIONS WHERE** specifies the conditional expression that determines which rows to restore to the table. This option can only be used if the following conditions are true:
 - The object is an individual table rather than a database.
 - The source and target tables have a defined **PARTITIONS BY** expression.
 - The restore is an all-AMP restore.
 - The table is excluded from the database object (via **EXCLUDE TABLES**) if the table belongs to a database object that is specified in the **RESTORE** script.
- **LOG WHERE** allows you to conditionally insert (log) archived rows that fall outside the partitions specified by the **PARTITIONS WHERE** conditional expression into a Teradata-generated error table. Teradata ARC inserts into the error table any rows that both match the **LOG WHERE** conditional expression and fall outside the partitions specified by the **PARTITIONS WHERE** conditional expression.

Error Table Name and Structure

The restore/copy process for selected partitions creates Teradata-generated error tables. Rows are stored in error tables for the following reasons:

- A row that is in the range of selected partitions being restored fails the target table integrity checks
- A user does not want to restore a row but it satisfies the **LOG WHERE** condition
- An error occurs while evaluating these conditions for a row

The restore/copy process for selected partitions creates an error table with the following characteristics:

- If a database name is not specified (either in the **ERRORDB** option or as a database qualifier of the error table name in the **ERRORTABLES** option), the error table resides in the same database as the target table being restored or copied; otherwise, the error table resides in the specified database.
- If the error table name is not specified (in the **ERRORTABLES** option), the error table has the name *RS_targettablename* truncated on the right as needed to 30 characters; if truncated, a warning message is issued. For example, if restoring table TB1, the default name for the error table is RS_TB1.
- If a table has the same name as an existing error table in a database, an error occurs. Before the restore/copy job can be submitted, this table must be dropped, renamed, or another table or database name specified. Review the contents of the existing table to determine whether to drop it:
 - If it is an error table, review for logged errors
 - If it is not an error table, consider using a different database or renaming for the error table

- If the error table is empty when the restore/copy job for the target table finishes, it is dropped.

Note: Do not share an error table between two or more restore/copy jobs. Each table targeted by a restore/copy job must have its own error table to ensure jobs run correctly.

Note: Do not drop the error table until the restore/copy job finishes. The error table must not exist for non-restart of restore/copy job. It must already exist for a restart of restore/copy job.

- An error table includes the following:
 - All columns have the same data types and attributes as the target table except for column-level integrity constraints and not null attributes.
 - DBCErrorCode has INTEGER data type, not null. It indicates the Teradata error causing the row to be inserted into the error table.
 - DBCOldROWID has BYTE(10) data type, not null. It indicates the internal partition number, row hash, and row uniqueness value of the row at the time the backup was created.
- An error table has the same FALLBACK or NO FALLBACK protection type as the associated target table.
- An error table is always a SET table even if the target table is a MULTiset table. Duplicate rows are discarded.
- The primary index of an error table is non-unique and is not partitioned. The primary index columns are identical to the primary index columns of the target table.
- An error table does not have any of the NOT NULL, table-level, or column-level integrity constraints of the target table. This is to ensure that rows can be inserted into the error table that might fail the integrity constraints.
- An error table does not have any secondary indexes or participate in any referential integrity relationships.

For example, assume that selected partitions of the following target table are being restored:

```
CREATE TABLE SalesHistory
  (storeid INTEGER NOT NULL,
   productid INTEGER NOT NULL CHECK(productid > 0),
   salesdate DATE FORMAT 'yyyy-mm-dd' NOT NULL,
   totalrevenue DECIMAL(13,2),
   totalsold INTEGER,
   note VARCHAR(256))
UNIQUE PRIMARY INDEX (storeid, productid, salesdate)
PARTITION BY RANGE_N(salesdate BETWEEN
  DATE '1997-01-01' AND DATE '2000-12-31'
  EACH INTERVAL '7' DAY)
INDEX (productid)
INDEX (storeid);
```

The restore job creates the following error table by default. The bold text points out pertinent differences from the target table definition.

```
CREATE SET TABLE RS_SalesHistory, FALLBACK
  (storeid INTEGER,
   productid INTEGER,
   salesdate DATE FORMAT 'yyyy-mm-dd',
   totalrevenue DECIMAL(13,2),
   totalsold INTEGER,
   note VARCHAR(256),
   DBCErrorCode INTEGER NOT NULL,
   DBCOldROWID BYTE(10) NOT NULL )
PRIMARY INDEX (storeid, productid, salesdate)
```

Restoring from a Dictionary Archive

If you restore a database from a dictionary archive created by specifying individual tables, only the dictionary rows for the specified tables are restored. [Table 4](#) alphabetically lists the dictionary rows that are restored by an all-AMPs data restore or a dictionary tables restore of a user database.

Table 4: User Database Data Dictionary Rows Restored

Table Rows	Description
Indexes	Definition of all indexed columns in the data tables and views
IndexName	Definition of named indexes in the table
TVM	Definition of all data tables, views, macros, stored procedures, triggers, and UDFs in the database
TVFields	Definition of all columns in the data tables, views, macros, stored procedure parameters, and UDFs.
TriggersTbl	Definition of all triggers in the database

[Table 5](#) lists the dictionary table rows that are restored by an all-AMPs restore of a specific user data table or journal table.

Table 5: User Table Dictionary Rows Restored

Tables Rows	Description
Indexes	Columns in tables that are indexes
IndexName	Indexes in the table that have names
TVM	Table names
TVFields	Columns for the table

Restoring from a Journal Archive

If you restore an all-AMPs archive of a journal table, the restored permanent journal goes to a different subtable than the journal currently writing the updates.

A restored journal overlays any change images that were restored to the same AMPs. During normal processing, the Teradata Database writes after-change images to a different processor than the one with the original data row. When you restore after-change images from an archive, these rows are written to the processor that contains the data row to which the change applies.

Restoring With All AMPs Online

You can restore even if AMPs are added to or deleted from the configuration after the archive.

Example

This example restores all databases with all AMPs online and assumes that none of the databases have journal table definitions. Journal tables must be dropped before running this job.

```
LOGON DBC, DBC;  
DELETE DATABASE (DBC) ALL, EXCLUDE (DBC);  
RESTORE DATA TABLES (DBC) ALL,  
    RELEASE LOCK,  
    FILE=ARCHIVE;  
LOGOFF;
```

The **DELETE DATABASE** statement is included so that only the tables in database DBC are present. If the **DELETE** statement is omitted, the restore cannot finish.

When the operation is complete, check the print file to verify that all AMPs involved in the restore process remained online. If one or more AMPs went offline, see [“Restoring With AMPs Offline” on page 48](#) for more information.

Restoring with a Specific-AMP Archive

Before you restore database DBC:

- 1 Reconfigure the system from a single AMP to the desired configuration.
- 2 Run the DBC Initialize Procedure (DIP) to initialize system views, macros, triggers, users, and error messages tables.

Example

This example demonstrates how to restore all databases from an all-AMPs archive and a specific-AMP archive. All AMPs are online for the restore, although AMPs may have been added to or deleted from the configuration since the archive was taken.

This example assumes that none of the databases have journal table definitions. Journal tables must be dropped before running this job.

```
LOGON DBC, DBC;  
DELETE DATABASE (DBC) ALL, EXCLUDE (DBC);  
RESTORE DATA TABLES (DBC),  
    FILE=ARCHIVE;  
RESTORE DATA TABLES (DBC) ALL,  
    EXCLUDE (DBC),  
NO BUILD,  
FILE=ARCHIVE;
```

```
RESTORE DATA TABLES (DBC) ALL,
    EXCLUDE (DBC),
NO BUILD,
FILE=ARCHIVEX;
BUILD DATA TABLES (DBC) ALL,
    EXCLUDE (DBC),
    RELEASE LOCK;
LOGOFF;
```

In this example, file ARCHIVE is an all-AMPs archive; ARCHIVEX is a single AMP archive.

When the restore is complete, check the print file to verify that all AMPs remained online throughout the process. If one or more AMPs went offline, see [“Restoring With AMPs Offline” on page 48](#) for more information.

To decrease the total time required to complete the restore process, specify the NO BUILD option for the restore operation, then submit a **BUILD** statement after the specific-AMP restore. In addition, unique secondary indexes remain valid on nonfallback tables.

Restoring Using the EXCLUDE Option

Before you restore database DBC:

- 1 Reconfigure the system from a single AMP to the desired configuration.
- 2 Run the DBC Initialize Procedure (DIP) to initialize system views, macros, triggers, users, and error messages tables.
- 3 Use the EXCLUDE option to:
 - Make one database, ADMIN, and its descendants available first; and make the other databases in the system available immediately thereafter.
 - Exclude one large database, U12, from being restored.

Example

In this example, an entire system is restored from an all-AMPs archive and one specific AMP archive. File ARCHIVE is an all-AMPs archive; ARCHIVEX is a single AMP archive:

```
LOGON DBC, DBC;
DELETE DATABASE (DBC) ALL, EXCLUDE (DBC);
RESTORE DATA TABLES (DBC),
    FILE=ARCHIVE;
RESTORE DATA TABLES (ADMIN) ALL,
NO BUILD,
FILE=ARCHIVE;
RESTORE DATA TABLES (ADMIN) ALL,
NO BUILD,
FILE=ARCHIVEX;
BUILD DATA TABLES (ADMIN) ALL,
    RELEASE LOCK;
RESTORE DATA TABLES (DBC) ALL,
EXCLUDE (ADMIN) ALL, (U12), (DBC),
NO BUILD,
FILE=ARCHIVE;
RESTORE DATA TABLES (DBC) ALL,
EXCLUDE (ADMIN) ALL, (U12), (DBC),
NO BUILD,
```

```
FILE=ARCHIVEX;  
BUILD DATA TABLES (DBC) ALL,  
    EXCLUDE (ADMIN) ALL, (U12), (DBC),  
    RELEASE LOCK;  
LOGOFF;
```

When the operation is complete, check the output to verify that all AMPs remained online during the restore.

Restoring With AMPs Offline

Restoring a Table with Fallback

If you restore a data table with fallback from an all-AMPs data tables or cluster archive while one or more AMPs are offline, Teradata ARC generates the information to restore the data on the offline AMPs when they return to operation. The system recovery process restores the offline AMPs when they return to online status.

Restoring a Table Without Fallback

If you restore a data table without fallback (or from a specific-AMP archive) while one or more AMPs that must also be restored are offline, restore the data table to each of the offline AMPs as soon as the AMPs come back online.

The exclusive utility lock is automatically placed on the data table for the all-AMPs restore and on each offline processor when the AMPs return to an online status. The lock allows Teradata ARC to finish the restoring process before the table can be accessed.

Unique Secondary Indexes

If you restore a table without fallback and with AMPs offline, unique secondary indexes defined for the table are invalidated. This prevents future updates to the table until you regenerate the unique secondary indexes either by dropping and recreating them or by using the **BUILD** statement.

Restoring Change Images

The Teradata Database always restores change images to the AMP that contains the data row, unless the row is from a single after-image journal. If a journal table contains change images for data tables with fallback, the database automatically creates the change images necessary for the offline AMPs when a recovery operation uses the restored journal.

If the journal table contains change images for tables without the fallback option, repeat the journal table restore for any offline AMPs when they return to online status. Any recovery activity using the restored journal must be repeated when the AMPs return to online status.

Restoring With One AMP Offline

The restore operation can occur even if AMPs have been added to or deleted from the configuration since the archive was taken.

To restore with an AMP offline:

- 1 Reconfigure the system from a single AMP to the desired configuration (optional).
- 2 Run the DBC Initialization Procedure (DIP) to initialize system views, macros, triggers, users, and error messages tables. See the software release cover letter for information on running DIP (optional).
- 3 Restore all AMPs by performing the operation described in [“Restoring With All AMPs Online” on page 46](#).
- 4 When the offline AMP is returned to online operation, perform a specific-AMP restore for that AMP.

Example

This example performs a specific-AMP restore for an AMP that was offline during the all-AMP restore.

```
LOGON DBC, DBC;
RESTORE NO FALLBACK TABLES (DBC) ALL,
      AMP=1,
      EXCLUDE (DBC),
      RELEASE LOCK,
      FILE=ARCHIVE;
LOGOFF;
```

When the operation is complete, check the print file to verify that all AMPs remained online during the restore. If one or more AMPs went offline, see [“Restoring With AMPs Offline” on page 48](#) for more information.

To decrease the total time required to complete the restore process, specify the NO BUILD option for the restore operation, then submit a **BUILD** statement after the specific AMP restore. In addition, unique secondary indexes remain valid on nonfallback tables.

Restoring Cluster Archives

When you restore all AMPs from a cluster archive, you must first restore the dictionary archive.

Restoring the dictionary archive does the following:

- Replaces the dictionary information for the tables, views, macros, stored procedures, triggers, and UDFs being restored in database DBC dictionary tables.
- Deletes any data (excluding journals) that belong to the database or tables being restored.
- Prevents fallback tables that are being restored from accepting Teradata SQL requests.
- Prevents nonfallback tables with indexes that are being restored from accepting Teradata SQL requests.

Remove tables from the restoring state with the **BUILD DATA TABLES** statement. This statement also revalidates any existing secondary indexes. The build operation should follow the restore of all cluster archives that represent the complete archive.

Restoring Cluster Archives in Parallel

If you choose to restore cluster archives in parallel, execute multiple jobs at the same time. Each of these jobs must have a different Teradata Database user identification because Teradata ARC does not allow concurrent restore operations by the same user.

Exclusive Locks

Each cluster restore requests exclusive locks within the various clusters. Therefore, the locks applied during the dictionary restore must be released prior to concurrent cluster restores.

Build Data Tables Operation

BUILD is run automatically for the primary data and all NUSIs. If the NO BUILD option is specified for a cluster restore and USI exists on one or more objects, a separate BUILD must be run on those objects after the entire cluster restore has completed. After you finish restoring all cluster archives, perform the build data tables operation against the tables that were restored. This generates the fallback copy of fallback tables and any indexes of fallback and nonfallback tables.

Rollforward Operation

If the build operation completes the recovery process and you are not performing a rollforward operation, you can release the locks.

If you need to do a rollforward, do it after the build operation using the guidelines described in [“Recovering Tables and Databases” on page 52](#).

To learn about the Rollforward operation, see [“ROLLFORWARD” on page 208](#).

Example

This example shows how to restore a cluster. The procedure is divided into three steps. Run Step 1 first, then run Steps 2 and 3 in parallel. Finally, run Step 4.

Job 1	<pre>LOGON USER1,USER1; RESTORE DICTIONARY TABLES (USERDB), RELEASE LOCK, FILE=ARCHIVE; LOGOFF;</pre>
Job 2 (run in parallel with Job 3)	<pre>LOGON USER1,USER1; RESTORE DATA TABLES (USERDB), CLUSTER=0, 1, FILE=ARCHIVE; LOGOFF;</pre>
Job 3 (run in parallel with Job 2)	<pre>LOGON USER2,USER2; RESTORE DATA TABLES (USERDB), CLUSTER=2, 3, FILE=ARCHIVE; LOGOFF;</pre>
Job 4	<pre>LOGON USER1,USER1; BUILD DATA TABLES (USERDB), RELEASE LOCK; LOGOFF;</pre>

Restoring After a Reconfiguration

If you have added or dropped AMPs since the archive was created, restore cluster and specific AMP archives to all AMPs. Because reconfiguration redistributes data among AMPs, rows in the archive do not distribute exactly to the AMPs from which the rows came.

Order of Restores

To restore a cluster archive to a reconfigured Teradata Database, first restore the dictionary archive and then individually restore each cluster archive to all AMPs. Only one restore operation is permitted at a time.

NO BUILD Option

When restoring cluster archives to all-AMPs, use the NO BUILD option on each cluster restore operation. If the NO BUILD option is omitted, Teradata ARC will print a warning and enable the option automatically. After all the cluster archives have been restored, a **BUILD** statement will need to be run.

If you restore an all-AMPs archive with nonfallback tables to all AMPs, and the all-AMPs archive is missing an AMP that has a specific-AMP archive, restore the all-AMPs archive using the NO BUILD option. In this case, nonfallback tables are not revalidated. Following completion of the restore of the all-AMPs archive, restore the specific-AMP archive to all AMPs. Then follow this specific-AMP archive restore with a build operation.

Logical vs. Physical Space

When restoring to a target system that is smaller than the source, the database space might be reported as a negative value when the operation finishes. This indicates you have run out of logical space, rather than physical space. If this occurs, reduce the perm space for some of the databases created on the system by doing the following:

- 1 Reduce the maximum perm space of user databases and rerun RESTORE.
- 2 If negative space is reported again, reset the Teradata Database, run UPDATESPACE to decrease the maximum perm space of user databases and rerun RESTORE.

Restoring with a Larger Number of AMPs

When running a complete restore (including database DBC) on a system with a larger number of AMPs, Teradata ARC can sometimes return an error that indicates the user database is out of space. This usually occurs only with databases that contain a large number of stored procedures or tables with poorly distributed data.

The reason for the error is that the perm space for a database is divided evenly among the AMPs on a system. When the target system has more AMPs, the perm space available on each AMP is lower. Tables with poorly distributed data might exceed the perm space allocation for one or more AMPs. If this error occurs, increase the perm space for the database and restart the restore operation.

Note: A reconfiguration that defines fewer disks will always be logically smaller than the original, even if it uses the same number of AMPs. To prevent the restore from resulting in

negative database space, reduce the maximum perm space of user databases *before* initiating a restore operation.

Restoring a Specific AMP

Each table has information about itself stored in a special row called a table header. Table headers are duplicated on all AMPs.

A request for a specific-AMP restore of a table is only allowed on tables without the fallback option. This type of restore is accepted only if a table header for the table already exists on the specified AMP. If a header does not exist, Teradata ARC rejects your request.

If you are restoring a single processor restoring after a disk failure, use the TABLE REBUILD utility to create the header for the table.

If a header already exists for the table, it must have the same structure as the header on the archive. If you change the definition of the table without rerunning the archive, Teradata ARC rejects the restore request.

Any data definition statement (for example, **MODIFY TABLE** or **DROP INDEX**) that you execute for a table also changes the structure (table header) of the table.

If the header structures match, Teradata ARC makes another check for invalid unique secondary indexes for the table. This check ensures that the specific AMP restore does not create a mismatch between data rows and secondary index rows.

If Teradata ARC finds any valid unique secondary indexes, it invalidates them automatically on all AMPs. An all-AMPs restore of such a table invalidates any unique secondary indexes on the table.

Restoring Encrypted Data from an Archive File

When the data in an encrypted archive file is restored, Teradata ARC automatically loads the data encryption module and algorithm that were used to originally encrypt the data. The data is decrypted after it is read from the archive file, and restored to the database as unencrypted data.

Recovering Tables and Databases

Teradata ARC provides functions to roll back or roll forward databases or tables from a journal table. To identify the databases and/or data tables to recover, use a Teradata ARC statement. If you specify multiple databases or data tables, they must all use the same journal table because a single recovery operation only uses change images from one journal.

A Teradata ARC statement also identifies whether the restored or current journal subtable is to be used in the recovery activity.

If you must recover data tables from more than one archived journal, perform the recovery operation in a series of three-step operations.

- 1 Restore an archived journal.
- 2 Recover the data tables using the restored journal.
- 3 Repeat steps 1 and 2 until all the tables are recovered.

Teradata ARC completes the requested recovery action as long as the table structure is the same as the structure of the change images. If the structures are not the same, Teradata ARC stops the requested recovery for the data table and returns an error message. When recovering multiple tables, the stopped recovery of one table does not affect recovery of the other tables.

The Teradata Database does not generate transient journal images during a rollback or rollforward operation. If the rollback or rollforward operation is not completed, the data tables being recovered are left in an unknown state.

If the rollback or rollforward is not completed because of a hardware failure, Teradata ARC automatically restarts the recovery operation.

If the rollback or rollforward operation is not completed because of a client failure, resubmit Teradata ARC with the RESTART option. If you release the HUT locks on the data tables prior to restarting and completing the Teradata ARC rollback or rollforward operation, the state of the data is unpredictable.

The Teradata Database does not generate permanent journal images during a rollback or rollforward operation. Consequently, a rollback operation might not be completed properly.

For example, assume:

- 1 A batch job updates a data table and creates both before- and after-images.
- 2 The batch job finishes.
- 3 Later (because of an error in the batch update) you roll the table back using the before-images.
- 4 A disk is replaced and you restore the most recent archive. You perform a rollforward to incorporate any changes that have occurred since the archive was created.

If the same journal that was used in the rollback described above is used in the rollforward, you might expect that the updates created by the batch job would not be reapplied since a rollback of the batch job had been done. Instead, the Teradata Database does not modify a journal table as the result of a recovery operation and the updates executed by the batch program are reapplied to the data tables during the rollforward.

Recovering With Offline AMPs

If you perform a recovery while AMPs are offline and the tables being recovered have fallback, the Teradata Database automatically generates the necessary information to recover the tables when the offline AMPs come back online. However, if the recovery includes tables without fallback, or if the recovery is from a cluster archive, you must restart recovery when the AMPs come back online.

Be especially careful of performing a rollforward operation on tables without fallback and with a single (local or remote) journal. If this type of operation uses the current journal as input with an offline AMP, the AMP that uses the offline AMP as its backup is *not* rolled

forward. However, if you specify the **RELEASE LOCK** option on the rollforward operation, then HUT locks are released on all AMPs except those that are down and nonfallback tables that have single after images and a down backup.

A similar option, the **BACKUP NOT DOWN** option, is available for the **RELEASE LOCK** statement.

Roll forward nonfallback tables with HUT locks remaining (because of a down backup AMP) when the down AMP is brought back online. Also, roll forward the down AMP when you bring it back online.

Recovering a Specific AMP

If you restore a nonfallback table with after-image journaling to a specific AMP after a disk failure, use a **ROLLFORWARD** statement followed by a **BUILD** statement of the nonfallback table.

If the nonfallback table has unique indexes, rollforward time may be improved by using the **PRIMARY DATA** option. This option instructs the rollforward process to skip unique secondary index change images in the journal. Because these indexes would be invalid from the specific-AMP restore operation, the **PRIMARY DATA** option might save a significant amount of I/O. Revalidate the indexes following the rollforward with the **BUILD** statement.

Copying Tables and Databases

Teradata ARC enables you to copy (or restore) a table or database to a different Teradata Database environment. Use the **COPY** statement to:

- Replace a table in a target database
- Create a table in a target database
- Move an archived file to a different Teradata Database other than the one from which the archive was made
- Move an archived file to the same Teradata Database from which the archive was made

Copy vs. Restore

The difference between *copy* and *restore* depends on the kind of operation being performed:

- A restore operation moves data from archived files back to the same Teradata Database from which it was archived or moves data to a different Teradata Database *so long as database DBC is already restored*.
- A copy operation moves data from an archived file to any Teradata Database *and* creates a new table if one does not already exist on the target database. When you copy selected partitions, the table must exist and be a table that was previously copied as a full-table copy.

Conditions for Using the COPY Statement

It is possible to create tables with different names using a **COPY** statement. For database-level copy operations, you can copy to a database with a different name.

To use the **COPY** statement, these conditions must be met:

- You must have restore access privileges on the target database or table to be able to use the **COPY** statement.
- A target database must exist to copy a database.
- If you copy a single table that does not exist on the target system, you must have both **CREATE TABLE** and **RESTORE** database access privileges for the target database.
- If the archived database contains a journal table that needs to be copied, the target database must also have a journal table.

Copying Database DBC

Do not use database DBC as the target database name in a **COPY** statement. You can use DBC as the source database in a **COPY FROM** statement, but you must specify a target database name that is different than DBC. This allows you to copy DBC from a source system to a different database on a target system. When used as the source database in a **COPY FROM** statement, database DBC is not linked to database SYSUDTLIB. Therefore, only database DBC is copied.

Copying Database SYSUDTLIB

Do not use database SYSUDTLIB as the target database name in a **COPY** statement. You can use SYSUDTLIB as the source database in a **COPY FROM** statement, but you must specify a target database name that is different than SYSUDTLIB. This allows you to copy SYSUDTLIB from a source system to a different database on a target system. When used as the source database in a **COPY FROM** statement, database SYSUDTLIB is not linked to database DBC. Therefore, only database SYSUDTLIB is copied.

Copying Data Table Archives

Always copy data tables before you copy any associated journal tables. When you copy a data table to a new environment, Teradata ARC creates a new table or replaces an existing table on the target Teradata Database. The **COPY** statement can only replace existing permanent journal tables; it cannot create permanent journal tables. For data table archives, note the following:

- If the target database does not have tables with the intended names, the copy operation creates them.
- If the target database has tables with the intended names, then they are replaced by the archived table data. Both the existing table data and table definition are replaced by the data from the archive.

When copying data tables, you can:

- Disable journaling
- Specify a journal table in a database different from the database that is receiving the table
- Change a fallback table to a nonfallback table

Copying Selected Partitions

Copying selected partitions in a PPI table works the same way as restoring selected partitions. For more information, see [“Archiving Selected Partitions of PPI Tables” on page 30](#).

HUT Locks in Copy Operations

Copy operations apply exclusive utility (HUT) locks on the object to be copied. Therefore, if you copy a full database from a complete database archive, Teradata ARC locks the entire database. Similarly, if you copy a single table from a table-level archive, Teradata ARC locks that table (but only that table). To copy selected partitions, Teradata ARC applies a HUT write lock.

Copying Large Objects (LOBs)

Teradata ARC supports copying tables that contain large object columns as long as the database systems are enabled for large object support and the copy is *not* for selected partitions. However, large object columns cannot be restored on a system that uses a hash function that is different than the one used for the archive. To copy an archive of selected partitions of LOBs, perform a full-table copy.

COPY Examples

The following examples illustrate data and journal table copies to all AMPs. Examples include copying:

- A table to a different configuration
- A data table to a new database
- A table to a new table or database
- A database to a new database
- Two databases with fallback
- A journal table from two archives

Copying a Table to a Different Configuration

An archived data table named `Personnel.Department` is copied to a different Teradata Database in this example:

```
COPY DATA TABLE (Personnel.Department)
  ,FILE = ARCHIVE;
```

In the example:

- The table has the same name in the target database that it had in the archived database, and it belongs to a database named `Personnel` on the target Teradata Database.

- If a database named Personnel does not exist on the target Teradata Database, Teradata ARC rejects the copy operation.
- If a table named Personnel.Department does not exist in the target database, Teradata ARC creates one.
- If the data table being restored has permanent journaling on the source system, it has permanent journaling on the target system as well.
- If Teradata ARC creates a new table for the copy operation, the target journal table is the default journal for database Personnel. Otherwise, the target journal table is the journal table associated with the table that is being replaced on the target system.
- If Teradata ARC creates a new table for the copy operation and if the Personnel database does not have a default journal, then Teradata ARC rejects the copy operation.

Copying a Data Table to a New Database

In this example, an archived data table is copied under a new database name to a different Teradata Database:

```
COPY DATA TABLE (Personnel.Department)
  (FROM (OldPersonnel), NO JOURNAL, NO FALLBACK)
  , FILE = ARCHIVE;
```

In this example:

- The new database name is Personnel; the old database name is OldPersonnel. In both databases, the table name is Department.
- If a database named Personnel does not exist in the target system, Teradata ARC rejects the copy operation.
- If a data table named Department does not exist in the Personnel database on the target system, Teradata ARC creates one with that name.
- The NO JOURNAL option indicates that you do not want this table to have permanent journaling in the target database.
- The NO FALLBACK option indicates that the new table is to be nonfallback on the target system, even if it was a fallback table on the archived system.

Copying a Table to a New Table or Database

An archived data table is copied under new database and table names to a different Teradata Database in this example:

```
COPY DATA TABLE (Personnel.Department)
  (FROM (OldPersonnel.Dept)
  , WITH JOURNAL TABLE = PersonnelJnlDB.PermanentJournal
  , NO FALLBACK)
  , FILE = ARCHIVE;
```

In the example:

- The new database name is Personnel; the old database name is OldPersonnel. The new data table name is Department; the old data table name is Dept.
- If a database named Personnel does not exist in the target system, Teradata ARC rejects the copy operation.

- If a data table named Department does not exist in the Personnel database on the target system, Teradata ARC creates one with that name.
- The WITH JOURNAL option indicates to carry journaling over from the old Teradata Database to the new Teradata Database for this table. In the example, the user copies journal images into the permanent journal table named PersonnelJnlDB.PermanentJournal.
- The NO FALLBACK option indicates that the new table is to be nonfallback on the target system, even if it was a fallback table on the archived system.
- If the table was nonfallback on the archive, then it retains its journaling options on the target system (that is, the journal options do not change).
- If the table was fallback on the archive, then its images are dual on the target system.

Copying a Database to a New Database

In this example, an archived database is copied with a new name to a different Teradata Database:

```
COPY DATA TABLE (Personnel)
  (FROM (OldPersonnel)
   ,WITH JOURNAL TABLE = PersonnelJnlDB.PermanentJournal
   ,NO FALLBACK)
  ,FILE = ARCHIVE;
```

In the example:

- The new database name is Personnel; the old database name is OldPersonnel.
- If a database named Personnel does not exist in the target system, Teradata ARC rejects the copy operation.
- The WITH JOURNAL option indicates to carry journaling over from the old Teradata Database to the new Teradata Database for any tables in the copied database that have journaling enabled on the archived system. In the example, the user copies journal images into the permanent journal table named PersonnelJnlDB.PermanentJournal. Although the tables are copied with the specified journal, the target database default does not change.
- The NO FALLBACK option indicates that the new table is to be nonfallback on the target system, even if it was a fallback table on the archived system.
- If the table was nonfallback on the archive, then it retains its journaling options on the target system (that is, the journaling options do not change).
- If the table was fallback on the archive, then its images are dual on the target system.

Although Teradata ARC copies the tables as nonfallback, the default for the target database does not change. For example, if the database named Personnel is defined with fallback, then the database keeps that default after the copy operation is complete. The NO FALLBACK option applies only to the tables within the archive.

Copying Two Databases With Fallback

Here, two databases are copied that have different fallback attributes:

```
COPY DATA TABLE (Personnel)
  (FROM (OldPersonnel), NO FALLBACK)
  , (Finance) (NO JOURNAL)
  , FILE = ARCHIVE;
```

In the example:

- The database named Personnel is restored from an archived database named OldPersonnel with all tables defined as nonfallback after the copy operation.
- The database named Finance is restored from an archived database also named Finance with all journaling associated with its data tables disabled.

Copying a Journal Table From Two Archives

This example illustrates how a journal table that has some nonfallback images spread across two separate archive files (because an AMP was down at the time of an all-AMPs archive operation) is restored:

```
COPY JOURNAL TABLE (PersonnelJnlDB.PermanentJournal)
  (APPLY TO (Personnel.Employee, Personnel.Department))
  , NO BUILD
  , FILE = ARCHIVE1;
COPY JOURNAL TABLE (PersonnelJnlDB.PermanentJournal)
  (APPLY TO (Personnel.Employee, Personnel.Department))
  , FILE = ARCHIVE2;
```

When restoring to a different configuration, Teradata ARC redistributes journal images among all the AMPs. Teradata ARC builds the journal only after it completes the all-AMPs data table archive operation.

To perform this operation:

- Copy the all-AMPs archive and specify the NO BUILD option.
- Copy the specific-AMP archive, but *do not* specify the NO BUILD option.

This operation is only allowed if the applicable data tables are already copied into the target system. In the example, the archive of all AMPs (except the AMP that was offline) is called ARCHIVE1. The archive of the AMP that was offline at the time of the main archive is called ARCHIVE2.

Teradata ARC copies only checkpoint journal images and images associated with the tables named Personnel.Employee and Personnel.Department to the target system. Teradata ARC discards all other journal images.

Copying Encrypted Data from an Archive File

When the data in an encrypted archive file is copied, Teradata ARC automatically loads the data encryption module and algorithm that was used to originally encrypt the data. The data is decrypted after it is read from the archive file, and copied to the database as unencrypted data.

Using Host Utility Locks

A Host Utility (HUT) lock, also referred to as a utility lock in this book, is the lock that Teradata ARC places when most Teradata ARC commands are executed. Exceptions are **CHECKPOINT**, **DELETE DATABASE**, and **DELETE JOURNAL**. When Teradata ARC places a HUT lock on a object, for example, with an **ARCHIVE** or **RESTORE** command, the following conditions apply:

- HUT locks are associated with the currently logged-on user who entered the statement, not with a job or transaction.
- HUT locks are placed only on the AMPs that are participating in a Teradata ARC operation.
- A HUT lock that is placed for a user on an object at one level *never* conflicts with another level of lock on the same object for the same user.

The ShowLocks utility, described in *Utilities*, shows the HUT locks that are currently applied to a Teradata Database.

Transaction vs. Utility Locks

From a blocking perspective, the behavior of a HUT lock is the same as a transaction lock. For example, a read lock prevents another job from claiming a write lock or exclusive lock. Conversely, a write lock prevents a read lock, write lock, or exclusive lock. An exclusive lock prevents any other type of lock. Conflicting HUT and transaction locks on an object block each other, just as if both were transaction locks or both were HUT locks.

The two differences between transaction locks and HUT locks are:

- **Permanence**
A transaction lock exists only for the duration of a transaction; after the transaction completes or is aborted, the lock is released. A HUT lock is more permanent. It is only removed if an explicit **RELEASE LOCK** command is issued, or if the locking operation (for example, execution of an **ARCHIVE** or **RESTORE** command) completes successfully and has the **RELEASE LOCK** option specified. The lock remains even if the command fails, the ARC job terminates, or if the Teradata system has a restart.
- **Scope**
A transaction lock has session scope, meaning that it applies to any command submitted by the session running the locking transaction. Another session running against the same object(s) must claim its own locks, even if it is running under the same user.
A HUT lock has user scope, meaning that it applies to any Teradata ARC operation that the user is performing on the object. A user must have only one HUT lock against an object at a time. The HUT lock allows any Teradata ARC operation from that user to access the locked object(s), but blocks other users from acquiring a conflicting HUT lock on the object(s). Additionally, the HUT lock blocks all conflicting transaction lock attempts on the object, even if the transaction locks are from the same user. Because there is only one HUT lock, a **RELEASE LOCK** operation for that user on an object always releases an existing HUT lock, even if other jobs for that user are still accessing the object.

Warning: Releasing a utility lock on a database or table that is being accessed by another Teradata ARC job could result in data corruption and unexpected errors from ARCMAN or the database.

HUT locks that are not released are automatically reinstated following a restart.

For information on transaction locking and processing, refer to *SQL Reference: Statement and Transaction Processing*.

Teradata ARC Locks During an Archive Operation

A Teradata ARC operation locks objects that are being archived.

- When archiving an entire database, Teradata ARC places the read utility lock at the database level before it archives any tables.
- When archiving individual tables, Teradata ARC places the read utility lock on each table before it archives that table.

If you specify the **RELEASE LOCK** option on the **ARCHIVE** statement, Teradata ARC releases the read utility lock when it successfully completes the archive of a database or table.

Therefore, only a full database archive creates consistency among tables archived from the same database.

Group Read Lock

You can specify a group read HUT lock for tables that are defined with an after-image journal option. If you are using the **GROUP READ LOCK** option to perform an online backup of a database-level object, define after-journaling for every table in the database. This restriction also applies to excluded tables; if a table is excluded as part of a database-level **GROUP READ LOCK** archive, it must still have an after-journal table defined so ARC can accept the **GROUP READ LOCK** option.

When you specify a group read HUT lock, the following events occur:

- The AMPs first place an access HUT lock on the entire affected table to prevent users from changing the definition of the table (that is, DDL changes) while the table is being archived.
- The AMPs then place a series of rolling read HUT locks on blocks of (approximately) 64,000 bytes of table data. Each AMP places a read HUT lock on a block of data while the AMP sends the data block across the IFP or PE to the client.

When a block transmission finishes, the AMP releases the read HUT lock on that block and places another read HUT lock on the next block of 64,000 bytes. The AMPs repeat this process until they archive the entire table.

Concurrent Transactions: Archives and Updates

When an archive with a group read HUT lock is finished, the archive may contain partial results of a concurrently executing update transaction. Consider the following example:

- 1 The archive operation reads row 1 and sends it to the client.
- 2 Transaction A reads and updates row 3.
- 3 The archive operation reads row 2 and sends it to the client.

- 4 Transaction A reads and updates row 1.
- 5 Transaction A ends, releasing its HUT lock on rows 1 and 3.
- 6 The archive operation reads row 3 and sends it to the client.

The archive described above contains:

- A copy of row 3 after an update by transaction A, and
- A copy of row 1 before its update by the transaction.

For this reason, you must define an after-image permanent journal for tables you want to archive using a group read HUT lock. After the archive finishes, archive the corresponding journal. The after-images on the journal archive provide a consistent view of any transaction that executes concurrently with the archive.

Archiving Using a GROUP READ LOCK

You can archive a table with a group read HUT lock at the same time users are updating it.

Tables archived with the group read HUT lock option do not archive with secondary indexes. You may rebuild secondary indexes after you restore any table archived with a group read HUT lock.

Restoring a Group Read Lock Archive

You must define tables that are archived under a group read HUT lock with an after-image journal in order to restore them. Always follow the restore of an archived data set taken under a group read HUT lock by performing a rollforward operation, using the after-image journal that was created during the archive. Rollforward ensures that the data in the tables is consistent. To learn more about the rollforward operation, see [“ROLLFORWARD” on page 208](#).

An archive taken under a group read HUT lock does not archive secondary indexes defined for the tables being archived. Therefore, you must rebuild these indexes after restoring such an archive. If you restore without specifying the NO BUILD option, Teradata ARC builds nonunique secondary indexes as part of the restore process. Teradata ARC does not rebuild unique secondary indexes. To learn more about the NO BUILD option, see [“NO BUILD Option” on page 51](#).

The Teradata ARC restore process marks as invalid all existing unique secondary indexes for the table being restored. Rebuild or drop all secondary indexes by doing one of the following:

- Issue a **BUILD DATA TABLES** statement.
- Drop and recreate the index.

Rebuild unique secondary indexes only after the rollforward of the data tables is complete.

Restoring a GROUP READ LOCK Archive of Selected Partitions

If you use the GROUP READ LOCK option for an archive of selected partitions, use the next procedure to perform the restore. To use this procedure, the structure version of the tables being restored must match the structure version of the tables stored on the backup file.

- 1 Restore or copy the selected partitions to the target table using the NO BUILD option.
- 2 Restore or copy journal tables of the journal backup associated with the time period of interest.
- 3 Roll forward the restored tables using the USE RESTORED JOURNAL and PRIMARY DATA options.
- 4 Submit a **BUILD DATA TABLES** statement to complete the build of the restored tables.

During this procedure, an "out-of-range" error condition can occur if the selected partitions for the restore are inconsistent with the change rows being rolled forward (that is, the journal contains rows outside of the selected partitions for the restore). ROLL processing continues on the indicated table and the out-of-range changes are made to the target table.

Caution: This procedure corrects the structural validity of the table but it does not help you understand the nature of the updates applied by the ROLL operation that fall outside of your selected partitions for the restore. It is possible that some significant partitions can be missed in the restore and only partially updated by the out-of-range changes. Carefully audit the entire recovery process to ensure that no data is missing. If you suspect that data is missing, repeat the above procedure for the affected partitions.

Teradata ARC Locks During a Restore Operation

A restore operation uses an exclusive HUT lock when data tables are restored.

- To restore an entire database, Teradata ARC places the exclusive lock at the database level.
- To restore selected tables, Teradata ARC places the exclusive lock on each table before it starts to restore the table.
- To restore (or copy) selected partitions of PPI tables, Teradata ARC uses a write lock.
- To restore selected partitions of PPI tables with GROUP READ LOCK, the entire table locks rather than a part of the table.

Restoring Tables

If you specify the RELEASE LOCK keyword in the **RESTORE** statement and that statement specifies that selected tables are to be restored, Teradata ARC releases the exclusive HUT lock on each table as soon as it finishes restoring it.

Restoring a Journal Table

When you restore a journal table, Teradata ARC places a write HUT lock on the table. The restored journal table and the journal table that is recording changed rows are different physical tables. Consequently, the write HUT lock exerted by a journal table restore does not prevent change images from recording to the journal table. The write HUT lock only prevents concurrent restore operations to the same journal.

Locks Associated with Other Operations

Teradata also uses the following locks.

Build Operation Exclusive Utility Lock

A build operation places an exclusive utility lock on databases where all tables are being built. Locks are applied only to the tables that are being built.

Checkpoint Operation Transaction Locks

The **CHECKPOINT** statement places *transaction* locks rather than utility (HUT) locks. Transaction locks are automatically released when the checkpoint operation terminates. A checkpoint operation places:

- A write lock on the journal table for which checkpoints are being set.
- Read locks on the data tables that may contribute change images to the journal table for which checkpoints are being set.

The operation places these locks and then writes the checkpoint. After the operation writes the checkpoint, Teradata ARC releases the locks.

Note: If you execute a checkpoint operation to mark the journal for later archiving using **CHECKPOINT** with the **SAVE** option, the AMP places an access lock on the data tables instead of a read lock.

Rollback/Rollforward Operations Exclusive and Read Utility Locks

Rollback and rollforward operations place these utility (HUT) locks:

- Exclusive locks on the tables to be recovered.
- A read utility lock on the journal table that is input to the recovery operation.

A journal table with a read utility lock can still log change images. The read utility lock on the journal table *does* prevent concurrent Teradata ARC statements, such as **DELETE JOURNAL**.

Delete Journal Operations Write Lock

The **DELETE JOURNAL** statement places *transaction* locks, not utility (HUT) locks. When the operation finishes, Teradata ARC automatically releases all transaction locks.

The **DELETE JOURNAL** statement places a write lock on the journal table to be deleted. Because the **DELETE JOURNAL** statement deletes only the saved journal, the active portion of the journal can still log change images during the delete operation. Other archive or recovery operations are not permitted.

Setting Up Journal Tables

Select the following journal options either at the database level or the table level. Specify single image or dual image journals.

These terms are used to describe journaling:

Note: Teradata ARC supports the generation of both before-image and after-image permanent journals, both of which can be either local or remote.

Table 6: Journal Options

Journal Options	Description
After image	The image of the row after changes have been made to it.
Before image	The image of the row before changes have been made to it.
Dual image	Two copies of the journal that are written to two different AMPs.
Primary AMP	The AMP on which the primary copy of a row resides.
Fallback AMP	The AMP on which the copy of a row resides. The Teradata Database distributes duplicate data rows to fallback processors by assigning the hash code of the row to a different AMP in the same cluster.
Backup AMP	The AMP on which copies of journal rows for nonfallback tables reside, including journals with single after-images, one copy of the dual after images, and one copy of the dual before images. Backup AMPs differ from fallback AMPs because journal images are not distributed to backup AMPs through a hashing algorithm. Instead, all images for one AMP go to a single backup. The backup for each AMP is always in the same cluster. For example, if AMPs A, B and C are in the same cluster, A backs up B, B backs up C and C backs up A.

Location of Change Data

Teradata ARC places change images in whatever journal table you define. The table can be in the same database as the data tables or it can be in another database.

Each database can contain only *one* journal table, but any journal table and the data tables that use it can reside in the same or in different databases. Set up journaling to have a *single* journal table for *all* data tables in the Teradata Database, a *separate* journal table for *each* table within a database, or a combination of these two.

If a data table does not have fallback protection, Teradata ARC always writes its after images to another AMP (backup AMP) in the same cluster as the one containing the data being changed.

[Table 7](#) shows where Teradata ARC writes journal change rows based on all the possible combinations of table protection and the specified journal option.

Table 7: Journal Change Row Location

Journal Option	Fallback	Location of Change Data
After	Yes	Primary and Fallback AMPs
Before	Yes	Primary and Fallback AMPs
Dual After	Yes	Primary and Fallback AMPs
Dual Before	Yes	Primary and Fallback AMPs
After	No	Backup AMP
Local After	No	Primary AMP
Before	No	Primary AMP
Dual After	No	Primary and Backup AMP
Dual Before	No	Primary and Backup AMP

If you specify the dual option, Teradata ARC writes an after-image on the primary and backup AMP. For a fallback table with a single after-image journal, Teradata ARC writes a journal row to the primary and fallback AMP. Teradata ARC also writes single before-images for nonfallback tables to the same processor as the data row being changed.

Subtables

Teradata Database maintains journals in tables similar to data tables. Each journal table consists of active, saved and restored subtables. The active and saved subtables are the current journal. Teradata ARC appends change images from updates to a data table to the active subtable. Teradata ARC also writes rows marking a checkpoint on the journal to the active subtable.

Before you archive a journal table, use the **CHECKPOINT** statement with the **SAVE** option. The checkpoint operation logically terminates the active portion of the journal and appends it to the current saved portion. Archive this saved portion to the client or delete it from the Teradata Database.

When you restore an archived journal to the Teradata Database, Teradata ARC places change images in the restored subtable. For more efficient recovery operations, Teradata ARC always places the change images on the processor they apply to, instead of placing them on a backup processor. You can delete a restored journal subtable with the **DELETE JOURNAL** table statement.

Roll operations can use either the current journal or the restored journal. When you specify the current journal, Teradata ARC automatically uses both the active and saved subtables of the journal table.

Local Journaling

Teradata ARC allows you to specify whether single after-image journal rows for nonfallback data tables are written on the same AMP as the changed data rows (called function called *local* single after-image journaling) or written to another AMP in the cluster (*remote* single after-image journal).

Use the local single after-image journal only with nonfallback tables. It is specified with the JOURNAL option in these SQL statements:

- **CREATE DATABASE**
- **CREATE USER**
- **MODIFY DATABASE**
- **MODIFY USER**
- **CREATE TABLE**
- **ALTER TABLE**

The system dictionary table/view columns describing journaling types are modified to be able to describe LOCAL single after-image journaling.

Teradata ARC rules for local single after-image journal and single before-image journal are the same except that a single local after-image journal is used with ROLLFORWARD only and single before-image journal is used with ROLLBACK only.

Local single after-image journaling reduces AMP path length. Without file system fault isolation support, the recoverability of user data tables against certain type of software errors with local single after-image journaling is less efficient than recovering with remote single after-image journaling.

Archiving Journal Tables

Example

In this example, journal tables for database PJ are archived to a file with a filename of ARCHIV1.

```
LOGON P/PJ,PJ;
CHECKPOINT (PJ) ALL, WITH SAVE;
ARCHIVE JOURNAL TABLES (PJ) ALL,
    RELEASE LOCK,
    FILE=ARCHIV1;
LOGOFF;
```

Journal Impact on Recovery

If a logical disk is damaged, data tables with local single after-image journaling might not be recovered beyond the last archived data. On the other hand, data tables with remote single after-image journaling might be recovered to the most recent transaction.

While the risks of cylinder corruption are minimal, it is possible for a cylinder containing both a user table and master journal, or a master index, to become corrupted. If this occurs, local journaling does not offer as much protection as remote journaling.

Controlling Journal Checkpoint Operations

A checkpoint places a marker at the chronological end of the active journal subtable. When the checkpoint is taken, the Teradata Database assigns an event number to it and returns that number as a response to the **CHECKPOINT** statement. In addition, you can supply a name with the **CHECKPOINT** statement and use it to reference the checkpoint in future operations.

During a checkpoint with a save operation, the Teradata Database logically appends the active subtable of the journal to the end of the saved subtable, then it automatically initiates a new active subtable.

You can archive or delete only the saved subtable of a journal. An **ARCHIVE JOURNAL TABLE** statement copies the saved subtable to the client. Use a **DELETE JOURNAL** table statement to physically purge the saved subtable.

Checkpoint Names

Checkpoint names must be unique within a journal subtable to avoid ambiguity. If checkpoint names are not unique, the Teradata Database uses one or both of the following:

- The last instance of a named checkpoint found in rollback operations
- The first instance of a named checkpoint found in rollforward operations.

Qualify a named checkpoint by the event number assigned to it by the Teradata Database.

Submitting a **CHECKPOINT** Statement

You can submit the **CHECKPOINT** statement both as a Teradata ARC statement and as a Teradata SQL statement (such as, through BTEQ or a user program). If you specify an SQL **CHECKPOINT** statement, the active journal is *not* saved.

When you enter a **CHECKPOINT** statement with the **SAVE** option, the Teradata Database creates the saved subtable. You can only enter this option through the Teradata ARC form of the **CHECKPOINT** statement.

Checkpoint and Locks

The checkpoint with a save operation can be taken with either a read lock or an access lock. If you use a read lock, the Teradata Database suspends update activity for all data tables that might write change images to the journal table for which checkpoints are being set. This action provides a clean point on the journal.

When you perform a save operation using an access lock, the Teradata Database takes all transactions that have written change images to the journal (which have not been committed) and treats them as though they started after the checkpoint was written.

Because you do not know how the Teradata Database treats particular transactions, a checkpoint with a save operation under an access lock is useful only for coordinating rollforward activities first from the restored journal and then from the current journal.

Completing a Checkpoint With Offline AMPs

You can issue a checkpoint statement while AMPs are offline. If AMPs are offline when you issue a checkpoint, Teradata ARC automatically generates a system log entry that takes the checkpoint on the offline AMPs as soon as they return to online status. The system startup process generates the checkpoint and requires no action by you.

In addition to generating the checkpoint entry in the journal, system recovery also updates the journal table with change images that were generated while the AMP was offline. These change images are either for fallback tables or dual image journals that should be on the AMP that was offline when the system took the checkpoint.

CHAPTER 3

Environment Variables

This chapter describes the variables used in Teradata ARC.

[Table 8](#) summarizes Teradata ARC environment variables. For more detail, refer to the sections that follow the table.

Table 8: Environment Variables

Variable	Description
ARCDFLT	The environment variable that points to the file containing the system-wide default parameters values, publicly accessible on the network.
ARCENV	The environment variable in which any valid Teradata ARC runtime parameters may be specified.
ARCENVX	Same as ARCENV, except that ARCENVX has the highest override priority. See ARCENV and ARCENVX in this chapter. Any runtime parameter set in ARCENVX is guaranteed to be used.

ARCDFLT

Purpose

ARCDFLT is the environment variable that points to the file that contains the default runtime parameters.

Usage Notes

Here is an example using ARCDFLT:

```
SET ARCDFLT=C:\TESTARC\CONFIG.ARC
```

The example above is equivalent to:

```
ARCMAN DEFAULT=C:\TESTARC\CONFIG.ARC
```

Almost all runtime parameters can be set as defaults in the file that is pointed to by ARCDFLT, so you can avoid long command-line arguments. ARCMAN loads the contents of the file pointed to by ARCDFLT when ARCMAN builds runtime parameters using these rules:

- If the path of a DEFAULT parameter file includes embedded spaces or special characters, enclose the path with single (') or double (") quote marks.
- Permanently set ARCDFLT using **Start > Control Panel > System > Environment**. But if a parameter is specified multiple times in different places, the override priority is:
 - a Parameters set in ARCENVX
 - b Actual runtime parameters on the command line
 - c Parameters set in ARCENV
 - d Parameters set in the default file pointed to by ARCDFLT

ARCENV and ARCVX

Purpose

ARCENV and ARCVX are environment variables in which any valid Teradata ARC runtime parameters can be specified.

Usage Notes

Here is an example using ARCENV and ARCVX:

```
SET ARCENV=WDIR=C:\TEMP\ARC\  
SET ARCVX=SESSIONS=20
```

Almost all runtime parameters can be set as defaults in ARCENV or ARCDFLT or in some combination of both. ARCMAN internally loads the contents of ARCENV, or the contents of the default file to which ARCDFLT points, when ARCMAN builds runtime parameters using these rules:

- If a parameter is specified multiple times in different places, the override priority is:
 - a Parameters set in ARCVX
 - b Actual runtime parameters on the command line
 - c Parameters set in ARCENV
 - d Parameters set in the default file pointed to by ARCDFLT
- You can specify runtime parameters in any order as long as PARM / DEFAULT is specified first.
- Commas or white spaces can be used as delimiters to separate ARCMAN runtime parameter syntax elements. If any of the runtime parameters require commas or white spaces in their values, the values must be delimited by double or single quotes:

```
WORKDIR='\my workdir'  
IOPARM='device=tape1 tapeid=100001 volset=weeklybackup'
```


CHAPTER 4

Runtime Parameters

You can specify Teradata ARC runtime parameters in any order as long as you specify PARM/DEFAULT first.

- For MVS systems, specify runtime parameters in ARCPARM or PARM on the EXEC card of the JCL.
- For VM systems, specify runtime parameters following ARCMAN in the EXEC file.

Table 9 summarizes Teradata ARC runtime parameters. For more detail, refer to the sections that follow the table.

Table 9: Runtime Parameters

Parameter	Platforms Supported	Description
CATALOG	All Platforms	Enables direct tape positioning for restore and copy operations.
CHARSETNAME	All Platforms	Enables support of Chinese/Korean character sets when connecting to the database.
CHECKPOINT	All Platforms	Places restart information in the restart log each time the specified number of data blocks are processed.
CHECKSUM	All Platforms	Allows the Teradata Database and Teradata ARC to verify data packets during ARCHIVE or RESTORE.
DATAENCRYPTION	All Platforms	Instructs Teradata ARC to use the payload encryption feature that is supported by Teradata Call-Level Interface (CLI).
DBSError	All Platforms	Allows the user to override the severity of specified error code. The new severity will determine how ARC handles the error.
DEFAULT	All Platforms	Defines the file path name that contains default options for Teradata ARC.
DEMODULE	Windows 2000/XP/Server 2003	Specifies a data extension module, which processes archive data prior to being sent to, or after being retrieved from, output media.
DEPARM	Windows 2000/XP/Server 2003	Specifies the parameters that are used with a data extension module (see DEMODULE).

Table 9: Runtime Parameters (continued)

Parameter	Platforms Supported	Description
ERRLOG (Specify the prefix NO to disable if previously enabled, even in the PARM file.)	All Platforms	Creates a file to store Teradata ARC execution errors.
FATAL	All Platforms	Allows you to set the program to reclassify a normally nonfatal error condition as fatal, and abend if the selected message is issued.
FILEDEF	All Platforms	Maps the internal name of a file to an external name, if specified.
HALT (Specify the prefix NO to disable if previously enabled, even in the PARM file.)	All Platforms	Terminates the current task if, during archive or recovery, the Teradata Database fails or restarts, and the new configuration differs from the configuration that preceded the failure.
HEX (Specify the prefix NO to disable if previously enabled, even in the PARM file.)	All Platforms	Displays object names in the standard output in hexadecimal notation.
IOMODULE	Linux, MP-RAS, Windows 2000/ XP/Server 2003	Specifies the name of an access module. Note: The value of IOMODULE is passed directly to the tape access module. ARCMAN does not interpret the value.
IOPARM	Linux, MP-RAS, Windows 2000/ XP/Server 2003	Specifies the initialization string for the access module named in IOMODULE. Note: The value of IOPARM is passed directly to the access module. ARCMAN does not interpret the value.
LOGON	All Platforms	Enables the runtime definition of the logon string.
LOGSKIPPED	All Platforms	Instructs Teradata ARC to log all of the tables that are skipped during an ARCHIVE operation.
OUTLOG	MP-RAS, Windows 2000/ XP/Server 2003	Duplicates standard (stdout) output to a file.
PARM	IBM MVS, VM	Specifies the frequently used runtime parameters are in a parameter file. Important: This parameter must be first when other parameters are specified.

Table 9: Runtime Parameters (continued)

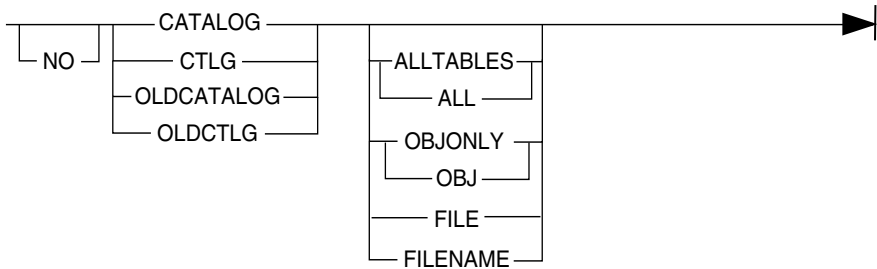
Parameter	Platforms Supported	Description
PAUSE (Specify the prefix NO to disable if previously enabled, even in the PARM file.)	All Platforms	Pauses execution if, during archive or recovery, the Teradata Database fails or restarts, and the new configuration differs from the configuration that preceded the failure.
PERFFILE	All Platforms	Provides a performance logging option for Teradata ARC to export performance data for job analysis.
RESTART (Specify the prefix NO to disable if previously enabled, even in the PARM file.)	All Platforms	Specifies the current operation is a restart of a previous operation that was interrupted by a client failure.
RESTARTLOG	All Platforms	Specifies the restart log name for ARCMAN.
SESSIONS	All Platforms	Specifies the number of Teradata sessions available for archive and recovery operations.
STARTAMP	All Platforms	Specifies a starting AMP for an all-AMPs archive rather than starting on a random AMP or starting with AMP0.
UEN (Utility Event Number)	MP-RAS, Windows 2000/ XP/Server 2003	Specifies the value that will be used to define %UEN% when in a RESTORE/COPY operation.
VERBOSE (Specify the prefix NO to disable if previously enabled, even in the PARM file.)	All Platforms	Sends Teradata ARC progress status information to the standard output device (for example, SYSPRINT for IBM platforms).
WORKDIR	MP-RAS, Windows 2000/ XP/Server 2003	Specifies the name of the working directory.

CATALOG

Purpose

CATALOG provides direct tape positioning for restore and copy operations when a table or a database is archived with the CATALOG runtime parameter enabled.

Syntax



2412a002

where

Syntax Element	Definition
CATALOG or CTLG	Enables CATALOG <i>at the table level</i> for ARCHIVE , RESTORE/COPY and/or ANALYZE statements.
CATALOGALLTABLES or CTLGALL	Synonyms for CATALOG.
CATALOGOBJONLY or CTLGOBJ	Enables CATALOG <i>at the object level</i> for ARCHIVE , RESTORE/COPY and/or ANALYZE statements.
CATALOGFILE or CTLGFILE	Enables CATALOG to write all catalog information to a file on the client machine. The default file name for the catalog is 'CATALOG' on mainframe systems, and 'CATALOG_<uen>.CTG' on other systems, where <uen> is the Utility Event Number for the archive job.
OLDCATALOG or OLDCTLG	When specified, along with CATALOG, enables use of the multi-table implementation of CATALOG, instead of the single-table method which does not create a new table for each ARCHIVE.
NOCATALOG or NOCTLG	Enables CATALOG for ARCHIVE , RESTORE/COPY and/or ANALYZE statements.
CATALOGFILENAME or CTLGFILE	Enables changing the file name for the catalog.
NOCATALOGFILE or NOCTLGFILE	Disables CATALOGFILE so that catalog information is not created during ARCHIVE, and not used during RESTORE or COPY.

The following syntax is not supported:

- CATALOGALL
- CTLGALLTABLES
- CATALOGOBJ
- CTLGOBJONLY

Usage Notes

CATALOG has two levels of granularity: OBJECT and ALL tables. If you specify CATALOG at the object level, the repositioning information is inserted into the CATALOG table for the database header record only.

Change the database in which the CATALOG table is created by adding this runtime parameter:

```
CATALOGDB=dbname
```

If you do not specify CATALOGDB, Teradata ARC uses \$ARC as the default CATALOG database.

Because catalog information is saved in the database, CATALOG enabled has a slight impact on performance. However, it is recommended that when archiving a database that contains many small or empty tables, disable CATALOG or, alternately, enable CATALOG at the object level.

Teradata ARC automatically excludes an active CATALOG database from its object list to avoid HUT lock conflicts. To archive, restore, or copy a CATALOG database, use NOCATALOG.

Automatic Activation of CATALOG

Teradata ARC automatically checks for the following macro before starting an archive operations if CATALOG is specified as a runtime parameter:

```
$ARC.ACTIVATE_CATALOG
```

If CATALOG is desired for all archive operations, create the following macro:

```
CREATE MACRO $ARC.ACTIVATE_CATALOG as (;);
```

Automatic activation of CATALOG only works for archive operations. Explicitly specify CATALOG for restore/copy operations.

The CATALOG Table

The default CATALOG database is \$ARC (or \$NETVAULT_CATALOG when using NetVault). It must be created and appropriate access rights must be granted before the first use of the CATALOG runtime parameter.

Catalog information for all archive jobs is kept in the one CATALOG table. To continue using the previous multi-table implementation of CATALOG, specify the OLDCATALOG parameter in addition to the CATALOG parameter. Also, you can still use old catalog tables

during RESTORE or COPY, even if you have not specified OLDCATALOG. ARCMAN searches for a table with the old naming style, and uses it if one exists.

Example

For UEN 1234, the following table is automatically created at the beginning of an archive operation:

```
$ARC.CATALOG_1234
```

If an error occurs when creating the CATALOG table, Teradata ARC terminates with a fatal error. Correct the error and resubmit the archive job.

During an archive operation, a CATALOG table row is inserted at the following instances:

- For each object, a database level CATALOG row is inserted at the end of the dictionary phase.
- For each table in the database, if the object level has not been specified, a table level CATALOG row is inserted at the end of the data phase.

The CATALOG tables can be used for detailed archive history in conjunction with the database DBC.RCEvent table. For information on the database DBC.RCEvent table, see the *Data Dictionary*.

Changing the Primary Index in the CATALOG Table

In previous versions of Teradata ARC, CATALOG tables were created with inefficient primary indexes that sometimes caused rows to be skewed on a limited number of AMPS. To correct this, follow these steps to change the primary index of an existing catalog table.

Note: If a new catalog table is created by Teradata ARC, these steps are not necessary.

- Create a new catalog table with the same DDL as the existing table:

```
CREATE TABLE CATALOGNEW AS CATALOG WITH NO DATA;
```

- Modify the new catalog table to use the correct primary index:

```
ALTER TABLE CATALOGNEW  
MODIFY PRIMARY INDEX (EVENTNUM, DATABASENAME, TABLENAME);
```

- Copy all data from the existing catalog data to the new catalog table:

```
INSERT INTO CATALOGNEW SELECT * FROM CATALOG;
```

- Remove the existing catalog table:

```
DROP TABLE CATALOG;
```

- Rename the new catalog table to “CATALOG”:

```
RENAME TABLE CATALOGNEW AS CATALOG;
```

CATALOG Operations

Archive

CATALOG no longer creates a table for each **ARCHIVE** statement; instead, CATALOG uses a single table named CATALOG. Multiple catalog tables are only created if you specify the OLDCATALOG parameter.

Restore

Use CATALOG in a restore operation only to selectively restore one or some databases or tables. To restore the majority of a tape or an entire tape, do not use CATALOG.

Because the UEN is saved in an archive header, ARCMAN knows which CATALOG table in the CATALOG database to access to retrieve catalog information for a restore operation. When ARCMAN is about to search for the database header, it retrieves the CATALOG row for the database and uses its repositioning information for direct tape repositioning.

These operations occur if CATALOG was enabled for the archive operation:

- If CATALOG was enabled at the table level, ARCMAN queries the CATALOG table for that specific table and uses the repositioning information for the table header search.
- If CATALOG was enabled at the object level at the time of the archive operation, then only database level repositioning is done using the catalog information; the tapes are scanned thereafter.

Copy

Use CATALOG in a copy operation only to selectively copy one or some databases or tables. If most or all of a tape is being copied, do not use CATALOG.

Copy operations to the same system from which data was archived work similarly to restore operations with respect to the use of catalog information. But to use CATALOG in a copy operation to a different system, you need to know the UEN of the tape. Obtain the UEN by using an **ANALYZE** statement or by reading the output from an **ARCHIVE** statement. With the UEN, you can archive and copy the corresponding CATALOG table from the source system to the target system before running a copy operation.

Mirrored Files

When two different data sets are archived by specifying File twice in an **ARCHIVE** statement, repositioning information is only saved in the CATALOG table for the primary, not the second (called the *mirrored*) data set.

If the mirrored data set is used in a restore operation with CATALOG, Teradata ARC detects that the mirrored data set is being used and automatically disables CATALOG.

Analyze

A CATALOG table can be generated offline using an **ANALYZE** statement that contains the CATALOG keyword, however, the **ANALYZE** statement does not recognize CATALOG specified as a runtime parameter. Tapes are scanned and the CATALOG rows are inserted into the CATALOG table for the database header records and the table header records in the tape set. After the CATALOG table is generated, it can be used in restore/copy operations.

Specifying CATALOG as a keyword in an **ANALYZE** statement is useful when a subset of objects must be restored or copied from a recataloged table set multiple times. However, for one-time operations, it is recommended that a **RESTORE** or **COPY** statement be used without specifying CATALOG as a runtime parameter because catalog generation using an **ANALYZE** statement requires scanning the entire tape set anyway.

CATALOGFILE

When you use the CATALOGFILE parameter, you can change the file name for the catalog with the CATALOGFILENAME command-line option. You can also disable CATALOGFILE by specifying NOCATALOGFILE or NOCTLGFILE. The effect is the same as NOCATALOG. Catalog information is not created during ARCHIVE, and is not used during RESTORE or COPY.

When using the CATALOG option of ANALYZE along with the CATALOGFILE / CATALOGFILENAME parameters, ANALYZE writes to both the CATALOG table on the database and the specified client-side file.

Note: In mainframe platforms (for example, MVS, VM), limit file names to eight characters.

NOCATALOG

NOCATALOG disables CATALOG. No CATALOG table is created during an archive operation; no catalog information is used during a restore/copy operation.

Dictionary Archive

The internal processing of a dictionary archive is equivalent to archiving a database with empty tables. Because there is no performance benefit for either archive or restore/copy operations for a dictionary archive, Teradata ARC automatically lowers the granularity of cataloging to the object level for dictionary archives. If you need table-level details for content/history purposes, generate them by running an ANALYZE of the archive file while using the CATALOG keyword.

CHARSETNAME

Purpose

The CHARSETNAME parameter adds support for the BIG5, GB, UTF-8, and Korean character sets to Teradata ARC. Use the CHARSETNAME parameter when you invoke the ARCMAN program at startup.

Syntax

CHARSETNAME _____ =name _____ ▶

2412a008

where

Syntax Element	Definition
<i>name</i>	Name of the selected character set

Usage Notes

It is possible to use Teradata ARC with a database that uses an alternate character set. Alternate character sets available for use with Teradata ARC include kanji ideographs, katakana, Korean, and traditional and simplified Chinese.

Use either CSNAME or CS as shorthand for CHARSETNAME.

When you specify the character set parameter, Teradata ARC uses the specified character set when connecting to the database. If you do not specify the option, Teradata ARC uses one of these default character sets for your client system:

- EBCDIC for channel-attached
- ASCII for network-attached

Available Character Sets

Use any of the following character sets to define the name parameter. Refer to *SQL Reference: Fundamentals* for information on defining your own character set.

Table 10: Teradata-Defined Character Sets

On this operating system...	Use this character set name...	For...
Channel-attached	EBCDIC	default
	KATAKANAEBCDIC	katakana
	KANJIEBCDIC5026	kanji 5026
	KANJIEBCDIC5035	kanji 5035
	TCHEBCDIC937_3IB	traditional Chinese
	SCHEBCDIC935_2IJ	simplified Chinese
	HANGULEBCDIC933_1II	Korean
Network-attached	ASCII	default
	UTF8	general
	KANJIEUC_0U	kanji EUC
	KANJISJIS_0S	kanji shift-JIS
	TCHBIG5_1R0	traditional Chinese
	SCHGB2312_1T0	simplified Chinese
	HANGULKSC5601_2R4	Korean

For compatibility with previous Teradata ARC versions, you can still individually specify the following character sets on the command line without using the CHARSETNAME parameter.

Channel-Attached Client Systems	Network-Attached Client Systems
EBCDIC	ASCII
KATAKANAEBCDIC	KANJIEUC_0U
KANJIEBCDIC5026_0I	KANJISJIS_0S
KANJIEBCDIC5035_0I	

It is not necessary to modify Teradata ARC jobs that used kanji character sets available in previous Teradata ARC versions.

Establishing a Character Set

You can establish a character set for a particular archive/recovery operation. However, once the character set is established, you cannot change it while Teradata ARC is executing an operation. When a character set is established, that character set accepts only Teradata ARC statements input in that character set.

Establish a character set in one of the following ways:

- At startup, invoke a parameter in JCL from MVS or in your VM EXEC. This user-specified character set overrides all other character sets previously specified or placed in default.
- Specify a character set value in the HSHSPB parameter module for IBM mainframes. This takes second precedence.
- If a character set is not user-specified or specified in HSHSPB, the default is the value in the system table, database DBC.Hosts.

If you rely on the database DBC.Hosts table for the default character set name, ensure that the initial logon from the channel-attached side is in EBCDIC. Otherwise, Teradata ARC does not know the default character set before logon.

Examples

On MVS, this JCL invokes ARCMAN:

```
//ARCCPY EXEC PGM=ARCMAN, PARM='SESSIONS=8 CHARSETNAME=KATAKANAEBDIC'
```

On VM/CMS, this EXEC invokes ARCMAN:

```
ARCMAN <SAMPLE.SYSIN.A SESSIONS=8 CHARSETNAME=KATAKANAEBDIC
```

Character Set Limitations

Using an alternate character set allows naming tables and databases using an extended set of characters. However, the internal representation of these extended characters depends on the session character set.

Although you can use Teradata ARC with a Teradata Database that uses an alternate character set, special care must be taken when archive and restore operations are used on a Teradata Database with an alternate character set:

- When archiving objects with names that use non-Latin characters, use the same character set defined using the multinational collation option for the database.
Multinational collation provides more culturally aware ordering of data in the database. If you use a different character set, Teradata ARC may have difficulty restoring the archive. To learn more about multinational collation, refer to *SQL Reference: Fundamentals*.
- To restore an archive made with an alternate character set, use the same character set used in the archive.
- You cannot restore an archive on a system that cannot handle the character sets used to create the archive.
- When the KATAKANAEBDIC character set is passed to Teradata ARC, all messages are uppercase because that character set does not support lowercase Latin letters.

When a Teradata ARC run begins while using an alternate character set, the software displays the following message, where *character_set_name* is the name of the character set defined in the runtime parameter:

```
CHARACTER SET IN USE: character_set_name
```

Note: Contact your system administrator or Teradata field support representative to learn more about the alternate character sets supported at your Teradata Database installation.

Limitations for Japanese Character Sets and Object Names

For certain character sets (including multibyte characters), Teradata ARC accepts non-Latin characters as part of an object name. An object name can be a database name, table name, user name, password, or checkpoint name in a Teradata ARC statement.

Teradata ARC also supports input and display of object names in hexadecimal notation. Teradata Database can store and display objects in hexadecimal format to allow object names to be archived and restored from one platform to another. To learn more about using non-Latin characters in object names, refer to *SQL Reference: Fundamentals*.

For example, to archive a database created on a UNIX platform from an IBM mainframe, you cannot specify the UNIX EUC character set on the IBM mainframe. Instead, specify the database name in the internal hexadecimal format understood by the Teradata Database.

The Teradata ARC allows you to specify object names in any of these formats:

- Without quotes. For example, for database DBC:

DBC

- With quotes. For example, for database DBC:

"DBC"

- External hexadecimal:

X'<object name in external hexadecimal format>'

This indicates that the specified hexadecimal string represents a name in the client (external) format. For example, for database DBC:

X'C4C2C3'

- Internal hexadecimal

'<object name in internal hexadecimal format>'XN

This indicates that the specified hexadecimal string represents a name in the Teradata Database internal format. For example, for database DBC:

'444243'XN

A Teradata ARC statement can contain any combination of object name notations. For example, the hexadecimal notation is a valid object name:

'4142'XN.TABLEONE

Limitations for Chinese and Korean Character Sets and Object Names

When using Chinese and Korean character sets on channel- and network-attached platforms, object names are limited to:

- A-Z, a-z
- 0-9
- Special characters such as \$ and _

Note: For more information on Chinese and Korean character set restrictions, refer to *SQL Reference: Fundamentals*.

Troubleshooting Character Set Use

It is recommended that you archive objects in the same character set class as the character set class from which they originated. It might not be possible to restore an object to a database that does not support alternate character sets. The name of a new table and a table that is part of the archive created using an alternate character set will not appear to be the same, and Teradata ARC considers the table as dropped during its restore of the archive.

If the Teradata Database fails to restore or copy a table or database that is archived, specify the following to perform a restore or copy operation:

- XN form for this object
- Session character set as the same character set as the one used to archive the object

If it is not possible to use the same character set for a restore operation as was used for the archive, specify the XN form of the object name for the particular table. The XN form will *not* be the normal internal form that was stored in the dictionary. Instead, it is the hexadecimal form of the object name stored in the archive. For example, to specify the XN name of an object in the **COPY/RESTORE** command:

```
COPY DATA TABLES
(mkw.'8ABF8E9A6575635F3235355F636F6C5F746162'xn)
(from ('616263'xn.'8ABF8E9A6575635F3235355F636F6C5F746162'xn)),
release lock,
FILE = ARCHIV1;
```

Teradata ARC might still not be able to locate the object on the archive because of collation differences between character sets and single-byte uppercasing applied to the object. Teradata ARC might fail to locate a object that was part of an archive when copying the table using ASCII or EBCDIC character sets because single-byte uppercasing was applied to the object.

If Teradata ARC cannot locate the object on the archive via the XN format, copy the object to a system that uses the character set of the archive, then archive it again. If the object name has non-Latin characters, rename the object to a valid Latin character name before archiving.

Sample JCL

The following sample line of code from the JCL illustrates how the name of a session character set is passed via the PARM runtime parameter to ARCMAN. In the example, the character set is "KANJI EBCDIC5035_01":

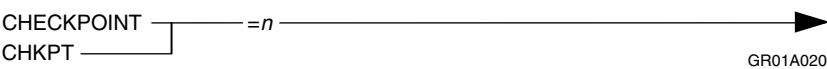
```
//COPY EXEC PGM=ARCMAN, PARM='KANJI EBCDIC5035_01'
```

CHECKPOINT

Purpose

The CHECKPOINT parameter saves restart information in the restart log at the intervals specified by this parameter.

Syntax



where

Syntax Element	Definition
<i>n</i>	<p>Number of data blocks processed before the checkpoint is taken.</p> <p>The maximum number of blocks is 10000000. Suffix K replaces three zeros (000). Setting CHECKPOINT to 0 disables ARC's GetPos request.</p> <p>If CHECKPOINT is not specified, the default number of blocks is 10000 for IBM and 32000 for Windows.</p>

Usage Notes

Both archive and restore/copy operations take checkpoints in the data phase. Every time the specified number of data blocks are processed, Teradata ARC saves the tape positioning and other processing information in the restart log. The frequency of checkpoint operations is controlled by the number of data blocks processed.

Because checkpoint operations causes I/Os and additional processing overhead, too many checkpoints might adversely impact performance. Each Teradata ARC data block contains up to 32K.

The CHECKPOINT parameter also controls the frequency of VERBOSE display, if active. To see the amount of data processed every minute with an approximate archive rate of 1MB/second, set the CHECKPOINT value between 2500 and 4500.

- Setting a checkpoint frequency too high (a low CHECKPOINT value) might impact performance.
- Setting a checkpoint frequency too low (a high CHECKPOINT value) causes Teradata ARC to reprocess a large number of data blocks in case of a restart.

Setting CHECKPOINT to 0

If CHECKPOINT is set to 0:

- Teradata ARC does not save positioning into the restart log. This means there is no restart/reconnect support. Some RESTORE/COPY operations also die with an error because no positioning information is saved.
- CATALOG is not supported and automatically disabled. Restart processing terminates with a fatal error.

For some tape drives, tape positioning information is not available or is very expensive. Setting CHECKPOINT to 0 allows more efficient processing in these cases.


Note: Setting CHECKPOINT to 0 is not the same as setting CHECKPOINT to a very large number. If CHECKPOINT is set to a very large number, Teradata ARC still saves some checkpoint information in the restart log, such as the beginning of database/table, and restart/reconnect *is* supported at database/table boundaries.

CHECKSUM

Purpose

The CHECKSUM parameter allows the Teradata Database and/or Teradata ARC to verify data packets during ARCHIVE or RESTORE.

Syntax

CHECKSUM _____ =*n* _____ 

2412a005

where

Syntax Element	Definition
<i>n</i>	<p>Checksum value, which can be one of the following:</p> <ul style="list-style-type: none">• 0 = checksum option is disabled.• 1 = the checksum is calculated by the Teradata Database.• 2 = the checksum is verified by Teradata ARC during ARCHIVE, and by both Teradata ARC and Teradata Database during RESTORE. <p>Note: CHECKSUM can also be used with the ANALYZE statement. If the VALIDATE option is used with ANALYZE, Teradata ARC validates the checksum for each data block in the archive.</p>

Usage Notes

Expect the following effects, depending on the value you set for CHECKSUM:

- Setting CHECKSUM=1 helps ensure data integrity during RESTORE, but this might cause a slight performance drop due to the fact that the checksum value for the data blocks is calculated only by the Teradata Database.
- Setting CHECKSUM=2 provides verification of data integrity during both ARCHIVE and RESTORE, but because both the Teradata Database and Teradata ARC calculate the checksum, this can result in a significant performance drop.

DATAENCRYPTION

Purpose

The DATAENCRYPTION parameter instructs Teradata ARC to use the payload encryption feature (for security) that is supported by Teradata Call-Level Interface (CLI).

Syntax

DATAENCRYPTION 
2412A024

Usage Notes

This parameter encrypts all Teradata ARC data that is sent across a network, but it does *not* encrypt data written to tape.

Caution: Because encryption and decryption require additional CPU resources, using this parameter significantly affects the backup and restoration of data being sent to and received by Teradata ARC.

For additional information about statements related to this parameter, see [“LOGDATA” on page 176](#) and [“LOGMECH” on page 181](#).

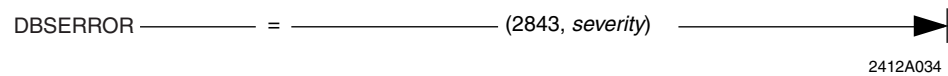
For more information about the encryption feature, see *Teradata Call-Level Interface Version 2 Reference for Network-Attached Systems*.

DBSERROR

Purpose

The DBSERROR parameter allows the specification of a Teradata Database error code, followed by the severity level of the error.

Syntax



where

Syntax Element	Definition
<i>severity</i>	Error severity level. Teradata ARC accepts any integer for the severity level on DBSERROR. For consistency, it is preferable to specify the values that Teradata ARC uses: 4 = warning 8 = non-fatal error 12 = fatal error 16 = internal error

Usage Notes

Currently, Teradata ARC allows only Teradata Database error code 2843 when using the DBSERROR parameter. Specify the severity level for the 2843 error code.

Example

```
ARCMAIN DBSERROR=(2843,4)
```

In this example, error code 2843 is set to a warning.

DEFAULT

Purpose

The DEFAULT parameter defines the file path name that contains default options for Teradata ARC.

Syntax

DEFAULT *filename* 

KM01A008

where

Syntax Element	Definition
<i>filename</i>	Name of the file that contains Teradata ARC default options

Usage Notes

- If DEFAULT is used, it must be the first runtime parameter specified in the list of parameters on the command line or in the environment variables.
- If the file path of the DEFAULT parameter includes embedded spaces or special characters, enclose the path with single (') or double (") quote marks.
- If ARCDFLT is specified, or if DEFAULT= starts either ARCENV or the actual command-line argument, then the file pointed to by ARCDFLT or DEFAULT is read into memory and its content prepended to the command-line string.
- On IBM platforms, PARM is the runtime parameter that provides the analogous functionality to DEFAULT.

The following example illustrates the use of DEFAULT:

```
DEFAULT=C:\TESTARC\CONFIG.ARC
```

Default Configuration File

If you use certain runtime parameters every time Teradata ARC is invoked, save them in a default configuration file pointed to by the DEFAULT runtime parameter. Each line in the default configuration file can contain one or more parameters. Add comments after two semicolons. If two semicolons are encountered in a line, the remaining characters of the line are ignored.

For example, the following default configuration file called C:\TESTARC\CONFIG.ARC includes a comment in the file and characters that are ignored:

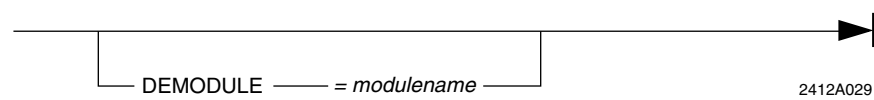
```
CATALOG;;enables direct positioning
VERBOSE
;;IOPARM= 'device=tape1 tapeid=1-5 volset=980731backup autoload=120'
FILEDEF=(archive,ARCHIVE_%UEN%.DAT)
CHKPT=1000
```

DEMODULE

Purpose

The DEMODULE parameter specifies the data extension module that processes data prior to being sent to, or after being retrieved from, output media.

Syntax



where

Syntax Element	Definition
<i>modulename</i>	<p>File name of the extension module to load.</p> <p>The file extension for the file name is optional. If the extension is not specified, the correct shared library extension for the platform is added (for example, <i>.dll</i> for Windows).</p> <p>Note: The DEPARM parameter is also required if DEMODULE is specified.</p>

Usage Notes

Note: This parameter is valid only on Windows platforms.

Data extension modules affect the processing of one or more ARC statements and must be specified before the ARC statement so that the data extension module will be active by the time processing of the ARC statement begins. The data extension module determines which ARC statements will be affected.

DEMODULE and DEPARM parameters are both required because they specify which data extension module is used, and the parameters that are applied to that data extension. After a data extension is enabled, it remains active until the end of the data operation (for example, an archive, restore, copy, or analyze operation).

Currently, the encryption data extension is supported. The data encryption module is used during an archive operation to encrypt the data read from the Teradata Database before the data is stored in the archive file. The encryption data extension is currently provided by

Protegrity. The Protegrity data encryption module, *pepbar.plm*, supports the AES128, AES256, and PANAMA data encryption algorithms.

When data in an encrypted archive file is restored, copied, or analyzed, the data encryption module and algorithm that was used to originally encrypt the data is automatically loaded by ARC to decrypt the data after it is read from the archive file. The data can then be processed as unencrypted data.

In the next example, the Protegrity data encryption module and the AES 128-bit algorithm are enabled. The single quotes enclosing the DEPARM value are optional. (In addition, the example also specifies eight Teradata Database sessions, direct tape positioning, and duplicate output to a file called *ARCALL.OUT*.)

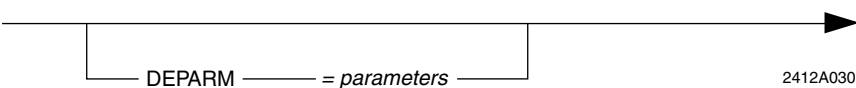
```
ARCMAN SESSIONS=8 CATALOG OUTLOG=ARCALL.OUT DEMODULE=PEPBAR.PLM  
DEPARM='ALGORITHM=AES128'
```


DEPARM

Purpose

The DEPARM parameter specifies the parameters of the data extension module. See [DEMODULE](#).

Syntax



where

Syntax Element	Definition
<i>parameters</i>	Parameters that are required for the module. Single quotes enclosing the parameters are optional. Note: The DEMODULE parameter is also required if DEPARM is specified.

Usage Notes

Note: This parameter is valid only on Windows platforms.

DEPARM and DEMODULE parameters are both required because they specify which data extension module is used, and the parameters that are applied to that data extension module. After a data extension module is enabled, it remains active until the end of the data operation (for example, an archive, restore, copy, or analyze operation).

Currently, the encryption data extension is supported. The data encryption module is used during an archive operation to encrypt the data read from the Teradata Database before the data is stored in the archive file. The encryption data extension is currently provided by Protegrity. The Protegrity data encryption module, *pepbar.plm*, supports the AES128, AES256, and PANAMA data encryption algorithms.

When data in an encrypted archive file is restored, copied, or analyzed, the data encryption module and algorithm that was used to originally encrypt the data is automatically loaded by ARC to decrypt the data after it is read from the archive file. The data can then be processed as unencrypted data.

In the next example, the Protegrity data encryption module and the AES 128-bit algorithm are enabled. The single quotes enclosing the DEPARM value are optional. (In addition, the example also specifies eight Teradata Database sessions, direct tape positioning, and duplicate output to a file called *ARCALL.OUT*.)


```
ARCMAN SESSIONS=8 CATALOG OUTLOG=ARCALL.OUT DEMODULE=PEPBAR.PLM  
DEPARM= 'ALGORITHM=AES128 '
```

ERRLOG

Purpose

The ERRLOG parameter duplicates all the error messages generated by Teradata ARC and stores them in an alternate file. But all error messages are still present in the standard output stream.

Syntax

ERRLOG `=name` 
GR01A023

where

Syntax Element	Definition
<i>name</i>	Name of the alternate error log file for storing these messages

Usage Notes

The use of the ERRLOG parameter depends on your operating system:

- On VM systems, *name* refers to an existing file definition (that is, FILEDEF) and is limited to eight characters. This limit does not apply to other platforms.
- On MVS systems, *name* refers to a DD name defined within the step that calls ARCMAN and is also limited to eight characters. This limit does not apply to other platforms.

Note: On all other platforms, *name* refers to a disk file in the current directory.

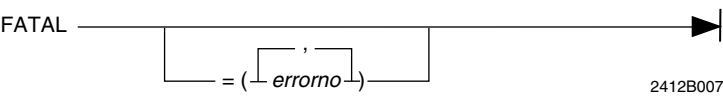
Specify NOERRLOG to disable the ERRLOG parameter.

FATAL

Purpose

The FATAL parameter provides a runtime option to the ARCMAN client that allows you to request that the program reclassify a normally nonfatal error condition as fatal, and abend if the selected message is issued.

Syntax



where

Syntax Element	Definition
<i>errorno</i>	Changes specified errors to a completion code of 12. This is a fatal condition which terminates ARC.

Usage Notes

FATAL changes the severity of any Teradata ARC error to FATAL, exit code = 12. For instance, if you specify that ARC0802 is a fatal error, Teradata ARC immediately exits at ARC0802.

In the next example, if an ARC0802 I/O error or ARC1218 warning message occurs, FATAL allows you to terminate Teradata ARC immediately rather than having the program run to completion. Either normally nonfatal error condition will be interpreted as fatal, and the program terminates:

```
ARCMAN FATAL=(802,1218)
```

The FATAL option allows you to stop the job immediately when one copy of the dump fails to ensure quick recovery action when only one good copy exists of the archive dump.

ARC1218 warns that a problem table will be skipped and the next table processed. If you determine that immediate termination is preferable to continuing, the FATAL parameter is invoked.

Delimiters, such as the equal sign, are mandatory.

FILEDEF

Purpose

The FILEDEF parameter maps the internal name of a file to an external name, if specified.

%UEN% enables ARCMAN to embed the UEN (Utility Event Number) in a filename

Syntax

FILEDEF = (*internal name* , '*external name*') —————▶

FILE ————

FDEF ————

KM01A005

where

Syntax Element	Definition
<i>external name</i>	External name of a file
<i>internal name</i>	Internal name of a file specified in the following Teradata ARC statements after the FILE keyword: <ul style="list-style-type: none"> • ANALYZE • ARCHIVE • RESTORE • COPY

Usage Notes

At execution, *internal name*, as specified in FILEDEF, is mapped to *external name*. But not all internal names need be defined. If the internal name is not defined in FILEDEF, the internal name itself is used as the external name.

Multiple FILEDEFs may be specified (one filename per FILEDEF). FILEDEFs do not override each other. Thus, there can be a list of FILEDEFs:

```
FILEDEF=(archive,ARCHIVE_%UEN%.DAT)
FILEDEF=(ARCHIVE1,C:\TESTARC\ARCHIVE1)
FILEDEF=(ARCHDICT,DICTIONARY_%UEN%)
```

If %UEN% is part of *external name*, then at execution of the Teradata ARC statement it is expanded to the actual UEN (Utility Event Number) or substituted with the value specified in the UEN runtime parameter in a **RESTORE/COPY** statement.

For example, if FILEDEF is defined as follows in an environment variable or in a default configuration file:

```
FILEDEF=(archive,ARCHIVE_%UEN%.DAT)
```

then in the following Teradata ARC script:

```
ARCHIVE DATA TABLES (DB), RELEASE LOCK, FILE=ARCHIVE
```


ARCHIVE is the internal name, ARCHIVE_%UEN%.DAT is the external name, and the final external name is something like ARCHIVE_1234.DAT, assuming that the UEN is 1234.

HALT

Purpose

The HALT parameter saves the current state into the restart log and terminates the current task if, during archive or recovery, the Teradata Database fails or restarts, and the new configuration differs from the configuration that preceded the failure.

Syntax

HALT 

GR01A024

Usage Notes

Use this parameter only when online configuration changes and non-fallback tables are involved. Processing is aborted when any table is non-fallback. If all tables are fallback, HALT is ignored. HALT affects the database as follows:

- If HALT is not specified, Teradata ARC displays a message and skips to the next table.
- If HALT and PAUSE are both specified, Teradata ARC displays an error message. Specify HALT or PAUSE, but not both.

Note: If you specify the HALT option and replace a disk after an archive or recovery operation has been started, the operation cannot be restarted from where it left off. The operation must be restarted from the beginning.

Specify NOHALT to disable the HALT parameter.

HEX

Purpose

The HEX parameter enables the display of object names in hexadecimal notation. This display is in internal Teradata Database hexadecimal format, except that the HEX display of object names from the **ANALYZE** statement is in the external client format (X'...').

Syntax

HEX 
GR01A025

Usage Notes

Use the HEX option to archive/restore table or database names that are created in a character set that is different from the currently specified character set.

If the HEX option is not used when object names are created in a character set that is not the currently specified character, the displayed object names might be unreadable or unprintable. The HEX option is retained across client and Teradata Database restarts.

Following are examples of the HEX parameter.

- From MVS:
`//ARCCPY EXEC PGM=ARCMAN, PARM='SESSIONS=8 HEX'`
- From VM/CMS:
`ARCMAN <SAMPLE.SYSIN.A SESSIONS=8 HEX`

Specify NOHEX to disable the HEX parameter.

IOMODULE

Purpose

The IOMODULE parameter specifies the name of an access module.

Syntax

IOMODULE _____ = *name* _____ 

KM01A001

where

Syntax Element	Definition
<i>name</i>	Name of an access module file

Usage Notes

The value of this parameter passes directly to the tape access module. ARCMAN does not interpret the value.

IOPARM

Purpose

The IOPARM parameter specifies the initialization string for the access module named in the IOMODULE runtime parameter.

Syntax



where

Syntax Element	Definition
'InitString'	Access module initialization string

Usage Notes

The value of this parameter passes directly to the tape access module. ARCMAN does not interpret the value.

LOGON

Purpose

The LOGON parameter enables the runtime definition of the logon string.

Syntax

LOGON = 'logon string' 

KM01A007

Usage Notes

If, for security reasons, you want to avoid defining a password in a Teradata ARC script, use the LOGON parameter to instead define a logon string at execution time.

If you specify the \$LOGON token in the **LOGON** statement, the logon string is replaced at execution.

Delimiters are mandatory.

Example

If you specify LOGON on the command line, as follows:

```
ARCMAN LOGON=' TDP6/USER1 , PWD1 '
```

and the Teradata ARC script contains:

```
LOGON $LOGON;
```

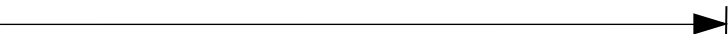
then \$LOGON is expanded to TPD6/USER1,PWD1 at runtime.

LOGSKIPPED

Purpose

The LOGSKIPPED parameter instructs Teradata ARC to log all of the tables that are skipped during an ARCHIVE operation.

Syntax

LOGSKIPPED 

2412A020

Usage Notes

The LOGSKIPPED parameter instructs Teradata ARC to log all of the tables that are skipped during an ARCHIVE operation. The tables are logged in a table named SkippedTables, which is created in the ARC catalog database (\$ARC by default, or \$NETVAULT_CATALOG when using NetVault).

Teradata ARC skips a table when the table is:

- Currently involved in a restore operation from a different instance of ARC
- Currently involved in a FastLoad or MultiLoad operation
- Missing a table header or a dictionary row, or is otherwise damaged

For each table that is skipped, ARC adds a row in the SkippedTables table. The columns included in the SkippedTables table are:

- EVENTNUM - Event number of the archive job where the table was skipped.
- DATABASENAME - Database containing the skipped table.
- TABLENAME - Name of the table that was skipped.
- ERRORTIME - TIMESTAMP value of when the error occurred on the table.
- ERRORCODE - Error code that caused Teradata ARC to skip the table.
- ERRORTTEXT - Error text for the corresponding error code.

Teradata ARC does not automatically retry tables that are added to the SkippedTables table. Therefore, ensure that these tables are separately archived when an error condition is fixed.

OUTLOG

Purpose

The OUTLOG parameter duplicates standard (stdout) output to a file.

Syntax

OUTLOG — = *name* —————▶

KM01A009

Usage Notes

If OUTLOG is specified, all output lines that go to standard (stdout) output are duplicated and saved in the file pointed to by OUTLOG.

Any existing file with the same name is overwritten.

In mainframe platforms (that is, MVS, VM), limit file names to eight characters.

PARM

Purpose

The PARM parameter identifies the file that contains frequently used runtime parameters.

Syntax



where

Syntax Element	Definition
<i>ddname</i>	<i>ddname</i> (VM/MVS platforms) of the file containing the frequently used runtime parameters

Usage Notes

The PARM runtime parameter saves frequently used runtime parameters in a separate parameter file. Its functionality is analogous to DEFAULT.

ARCMAN reads the PARM file, then builds the parameter list line and appends the actual parameter list to it. PARM or *PARM=ddname* must be the first parameter in the parameter list, meaning that PARM cannot appear in the middle of the parameter list.

On IBM platforms, if PARM is specified without a *ddname*, then the default *ddname* is *SYSARM*.

Use the following NO parameters to disable any previously specified runtime parameters, including those specified in PARM:

- NOHALT
- NOPAUSE
- NORESTART
- NOHEX
- NOVERBOSE
- NOERRLOG

It is possible to specify the SESSIONS and CHECKPOINT parameters more than once to override previous instances of these parameters. However, the last instance of the parameter is the value that is used. For example, if these runtime parameters are specified once in PARM, then one or more times in the parameter list, only the last instance of the parameter is used.

Example

The next example illustrates the use of PARM, including an instance when more than one parameter (in this case, SESSIONS) is specified in PARM:

```
//REST EXEC PGM=ARCMAN,PARM='SESSIONS=80 SESSIONS=40 VERBOSE'
```

In the example, 40 sessions are used, not 80, because SESSIONS=40 is the last instance of the parameter specified in PARM.

JCL for the Above Example

Following is sample JCL that includes a PARM file and a parameter list.

```
//REST      EXEC PGM=ARCMAN,PARM='PARM NOHALT CHECKPOINT=2000'
//*-----
//* A SEPARATE DATASET CAN BE USED AND SYSPARM JUST POINT TO IT.
//* SYSPARM IS EXPLICITLY SHOWN HERE FOR ILLUSTRATION PURPOSE.
//SYSPARM DD *
VERBOSE
CHECKPOINT=5000
SESSIONS=80
HALT
/*
//DBCLOG    DD DSN=&&T86CLOG,DISP=(NEW,PASS),
//          UNIT=SCR,SPACE=(TRK,(8,8),RLSE)
//ARCHIVE   DD DSN=TEST.TEST.TEST,
//          DISP=(OLD,KEEP,KEEP)
//SYSPRINT DD SYSOUT=*
//SYSTEM    DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//SYSIN     DD *
LOGON TDQA/TEST,TEST;
COPY DATA TABLES (TEST)
      ,RELEASE LOCK
      ,DDNAME=ARCHIVE;
LOGOFF;
/*
```

Note: For both MVS or VM, DDNAME and the FILE keyword are interchangeable. For example, *DDNAME=ARCHIVE* is the same as *FILE=ARCHIVE*.

Sample Output


```
04/29/97 12:15:57      ****      ****      ****      DATABASE COMPUTER
04/29/97 12:15:57      *      *      *      *      *
04/29/97 12:15:57      *      *      ****      *      PROGRAM: ARCMAN
04/29/97 12:15:57      *      *      *      *      *      RELEASE: 06.01.00
04/29/97 12:15:57      ****      ****      ****      BUILD: 9702-16 (Apr      25
1997)
04/29/97 12:15:57      Reading runtime parameters from SYSPARM
04/29/97 12:15:57
04/29/97 12:15:58      RUNTIME PARAMETERS SPECIFIED:
04/29/97 12:15:58      VERBOSE CHECKPOINT=5000 SESSIONS=80 HALT
                           NOHALT CHECKPOINT=2000
04/29/97 12:15:58
04/29/97 12:16:00      PARAMETERS IN USE:
04/29/97 12:16:00
04/29/97 12:16:00      SESSIONS 80
04/29/97 12:16:00      VERBOSE LEVEL 1
04/29/97 12:16:00      CHECKPOINT FREQUENCY CHANGED TO 2000
04/29/97 12:16:00
04/29/97 12:16:01      CHARACTER SET IN USE: EBCDIC
```


PAUSE

Purpose

The PAUSE parameter saves the current state into the restart log and pauses execution if, during archive or recovery, the Teradata Database fails or restarts and the new configuration differs from the configuration that preceded the failure.

Syntax

PAUSE 
GR01A026

Usage Notes

Use this parameter for non-fallback tables only.

PAUSE causes Teradata ARC to wait for five minutes, then checks the current configuration to see if it has returned to its previous state. PAUSE remains in its delay loop until the Teradata Database either returns to its previous state or a time limit is exceeded:

- If the AMPs are in the original configuration, Teradata ARC continues processing the archive or restore operation from where it left off.
- If the configuration still differs, Teradata ARC pauses for another five minutes and repeats the process described above.

Teradata ARC repeats the five minute pause and check process 48 times (about four hours). If after 48 times the configuration is still different, Teradata ARC saves the current state into the restart log and terminates the current task.

- If PAUSE is not specified with non-fallback tables, the archive is aborted and Teradata ARC skips to the next table.
- If HALT and PAUSE are both specified, Teradata ARC displays an error message. Either HALT or PAUSE may be specified, but not both.

In the VM environment, use the **CP SET TIMER REAL** command to ensure the SLEEP function works when using PAUSE.

Note: If you specify the PAUSE option and replace a disk after an archive or recovery operation is started, the operation cannot be restarted from where it left off. The operation must be restarted from the beginning.


Specify NOPAUSE to disable the PAUSE parameter.

PERFFILE

Purpose

The PERFFILE parameter produces a log of Teradata ARC performance data as a source for job performance monitoring, analysis, and debugging.

Syntax

PERFFLE  *=PLOG*

2412a009

Each line of the performance file contains a two-character record type followed by the data provided for this type, as shown in the next table.

Syntax Element	Definition
BG <timestamp> <IOPARM>	Beginning of performance log. <IOPARM> is the list of parameters given in the IOPARM command-line option (if any).
EN <timestamp> <exitcode>	Indicates the end of the Teradata ARC job. <exitcode> is the final exit code returned by ARC.
ER <timestamp> <exitcode> <errorcode> <string>	Indicates a warning, error, or failure by ARC. <exitcode> is the severity of the error (4 for warning, 8 for error, 12 for failure, and 16 for internal error). <errorcode> is the Teradata ARC or Teradata Database error code for the error. <string> is the string output by Teradata ARC indicating the error.
LG <timestamp> <logon-str>	Displays the logon time and the logon string given to Teradata ARC in the ARC script.
OE <timestamp> <TS-end> <objtype> <name>	Indicates the end time for the named object. <TS-end> is the end time for the object. <objtype> is the type of object (database-level, 'D', or table-level, 'T'). <name> is the name of the database (and table for table-level objects).
OS <timestamp> <objtype> <name>	Indicates the start time for the named object. <objtype> is the type of object (database-level, 'D', or table-level, 'T'). <name> is the name of the database (and table for table-level objects).

Syntax Element	Definition
ST <timestamp> <statement>	Displays the currently executing statement (from the ARC script).
TB<TS-begin><TS-build> <TS-end> <bytes> <name>	Indicates the output at the end of archiving or restoring each table. TS-begin is the time the table started. TS-build is the time the build process started. TS-end is the ending. <bytes> is the number of bytes of data archived or restored. <name> is the name of the database and table archived or restored.
VB <timestamp> <string>	Indicates that a verbose message is output by ARC. <string> is the message that is displayed.

Usage Notes

In mainframe platforms (that is, MVS, VM), limit file names to eight characters.

RESTART

Purpose

The RESTART parameter specifies that the current operation is a restart of a previous operation that was interrupted by a client failure.

Syntax

RESTART 
GR01A027

Usage Notes

In the case of restart, Teradata ARC uses the SESSIONS parameter that is specified the first time the utility is run. Do not change the input file between the initial execution and restart.

Specify NORESTART to disable the RESTART parameter.

For further information about dealing with client failures during an archive or recovery operation, see [Chapter 7: “Restarting Teradata ARC.”](#).

RESTARTLOG

Purpose

The RESTARTLOG parameter specifies the restart log name for ARCMAN.

Syntax

RESTARTLOG — = *filename* —————▶
RLOG —————

KM01A003

Usage Notes

The RESTARTLOG runtime option is available only on Windows and MP-RAS platforms. (The restart log name on IBM VM/MVS platforms cannot be changed by RESTARTLOG.)

Use the RESTARTLOG as follows:

- Teradata ARC adds the extension type RLG to the name of the file specified in RESTARTLOG. Therefore, do not use this extension in the name of the restart log.
- The restart log is created under the current directory or the working directory, if defined, unless the full path is specified.
- If RESTARTLOG is not specified, database DBCLOG is the default name of the restart log on the IBM platform. On other platforms, *yymmdd_hhmmss* is date and time, and *<n>* is the process ID in the following example of a default restart log name:
ARCLOGyymmdd_hhmmss_<n>.RLG.
- Teradata ARC does not automatically remove the restart log files after the successful completion, therefore clean the files periodically.
- Delimiters depend on the name of the restart log being specified.

Example

```
RESTARTLOG=RESTARTLOG
RLOG='C:\TESTARC\JOB12345'
```

SESSIONS

Purpose

The SESSIONS parameter specifies the number of Teradata Database sessions that are available for archive and recovery operations. This number does not include any additional sessions that might be required to control archive and recovery operations.

Syntax



where

Syntax Element	Definition
<i>nnn</i>	Number of data sessions. The default is 4.

Usage Notes

In general, the amount of system resources (that is, memory and processing power) that are required to support the archive or recovery increases with the number of sessions. The impact on a particular system depends on the specific configuration.

Teradata ARC uses two control sessions to control archive and recovery operations. The **LOGON** statement always connects these sessions no matter what type of operation being performed. Teradata ARC connects additional data sessions based on the number indicated in the SESSIONS parameter. These sessions are required for the parallel processing that occurs in archive and restore or copy operations.

If additional data sessions are required, Teradata ARC connects them at one time. Teradata ARC calculates the number of parallel sessions it can use, with maximum available being the number of sessions indicated with this parameter. Any connected sessions that are not actually used in the operation result in wasted system resources.

Table 11 indicates the data session requirements for archive and restore operations.

Table 11: Data Session Requirements

Operation	Data Session Requirements
All-AMPs archive	No more than one session per AMP.
Cluster or specific archive	Teradata ARC calculates the nearest multiple of online AMPs for the operation and assigns tasks evenly among the involved AMPs. For example, if you archive four AMPs with nine sessions specified, Teradata ARC actually uses eight sessions (two per AMP). The extra connected session wastes system resources since it is unused.
Restore/copy	Restore operations use all the sessions specified using the sessions parameter.

STARTAMP

Purpose

The STARTAMP parameter allows you to specify a starting AMP for every all-AMP ARCHIVE job.

Syntax



where

Syntax Element	Definition
<i>ampid</i>	Starting AMP for a single all-AMP ARCHIVE job

Usage Notes

Using STARTAMP during multiple all-AMP ARCHIVE jobs can increase archive performance. Specifying a different AMP for each job minimizes potential file contention. If STARTAMP is not used, Teradata ARC randomly selects a starting AMP for each archived table.

Example

```
STARTAMP=1
or
AMP=1
```


UEN (Utility Event Number)

Function

The UEN parameter specifies the value that defines %UEN% in a RESTORE/COPY operation. UEN is automatically generated during an archive.

Syntax

UEN — = *n* —————▶

KM01A006

where

Syntax Element	Definition
<i>n</i>	Utility event number, for example, UEN=12345

Usage Notes

In a **RESTORE/COPY** statement, %UEN%, if used, is expanded using the Utility Event Number specified in UEN.

Example

If FILEDEF is defined as follows in an environment variable or in a default configuration file:

```
FILEDEF=(archive,ARCHIVE_%UEN%.DAT)
```

then in the following Teradata ARC script, *ARCHIVE* is the internal name, and *ARCHIVE_%UEN%.DAT* is the external name:

```
ARCHIVE DATA TABLES (DB), RELEASE LOCK, FILE=ARCHIVE
```

The final external name of this ARC script is *ARCHIVE_1234.DAT*, assuming that the UEN is 1234. (The output of this archive displays the UEN.)

Therefore, if you run a restore with *UEN=1234* and use the above FILEDEF, then in the following Teradata ARC script, the actual external name will be *ARCHIVE_1234.DAT*:

```
RESTORE DATA TABLES (DB), RELEASE LOCK, FILE=ARCHIVE
```

VERBOSE

Purpose

The VERBOSE parameter directs Teradata ARC progress status information to the standard output device (for example, SYSPRINT for IBM platforms). The amount of status information sent is determined by the level specified in the parameter.

Syntax



where

Syntax Element	Definition
n	<p>Level of program status information sent to the standard output device, where n represents 2 or 3.</p> <p>VERBOSE without an n value provides the lowest level of status information.</p> <p>The amount of process status information sent to standard output increases with a higher n value.</p>

Usage Notes

Progress status report is displayed on a standard output device. On the IBM mainframe platform, the progress status is sent to the SYSPRINT DD with other regular Teradata ARC output lines.

VERBOSE provides three levels of progress status (1 through 3):

- Level 1 is specified as:
VERBOSE, VERB or VB
- Levels 2 through 3 are specified as:
Level 2 - VERBOSE2, VERB2 or VB2
Level 3 - VERBOSE3, VERB3 or VB3

While VERBOSE level 1 is designed for normal operation, the other two levels are for diagnosis and debugging. The performance of Teradata ARC is affected when these higher VERBOSE levels are used, especially if a low CHECKPOINT value is specified.

The output of VERBOSE is preceded by three dashes (---) for easy identification and filtering by other utilities.

For ARCHIVE and RESTORE/COPY operations, the amount of data processed in the data phase is displayed at each checkpoint. Teradata ARC internally takes checkpoints for every n number of data blocks processed for restart purposes. Therefore, it is possible to control the frequency of display of status information when VERBOSE is specified by the CHECKPOINT parameter setting. See [“CHECKPOINT” on page 88](#).

For the RESTORE/COPY operations, each build request is displayed in addition to the data processed before the request is sent to the Teradata Database. However, because the client has no control of the actual build execution, once the request is sent, Teradata ARC waits for the completion of the build request.

Specify NOVERBOSE to disable the VERBOSE parameter. The default is NOVERBOSE.

Example

Following is sample output that results from specifying VERBOSE level 1.

Caution: The organization and contents of informational progress report lines changes without notice with new releases of Teradata ARC. Therefore, do not develop scripts or programs that exactly copy this example.

```

04/29/97 10:40:38      ****      ****      ****      DATABASE COMPUTER
04/29/97 10:40:38      *    *    *    *    *
04/29/97 10:40:38      *    *    ****    *      PROGRAM: ARCMAN
04/29/97 10:40:38      *    *    *    *    *      RELEASE: 06.01.00
04/29/97 10:40:38      ****      ****      **** BUILD: 9702-16 (Apr 25 1997)
04/29/97 10:40:38
04/29/97 10:40:38      RUNTIME PARAMETERS SPECIFIED:
04/29/97 10:40:38      verbose checkpoint=5
04/29/97 10:40:38
04/29/97 10:40:40      PARAMETERS IN USE:
04/29/97 10:40:40
04/29/97 10:40:40      SESSIONS 4
04/29/97 10:40:40      VERBOSE LEVEL 1
04/29/97 10:40:40      CHECKPOINT FREQUENCY CHANGED TO 5
04/29/97 10:40:40
04/29/97 10:40:41      CHARACTER SET IN USE: EBCDIC
04/29/97 10:40:41      LOGON TDRU/JCK,
04/29/97 10:40:42      LOGGED ON      2 SESSIONS
04/29/97 10:40:42      STATEMENT COMPLETED
04/29/97 10:40:42
04/29/97 10:40:42      ARCHIVE data tables (jck),
04/29/97 10:40:42      release lock,
04/29/97 10:40:42      file=archive;
04/29/97 10:40:43      LOGGED ON      4 SESSIONS
04/29/97 10:40:44      UTILITY EVENT NUMBER - 79
04/29/97 10:40:44
04/29/97 10:40:44      ARCHIVING DATABASE "JCK"
04/29/97 10:40:54      --- Starting table "ACCESSRIGHTS"

```

```

04/29/97 10:40:56 --- CHKPT: 129,510 bytes, 2,184 rows received
04/29/97 10:40:57 --- CHKPT: 183,200 bytes, 3,094 rows received
04/29/97 10:40:58 --- CHKPT: 322,971 bytes, 5,463 rows received
04/29/97 10:40:59 --- CHKPT: 436,310 bytes, 7,384 rows received
04/29/97 10:41:00 --- CHKPT: 526,226 bytes, 8,908 rows received
04/29/97 10:41:01 --- CHKPT: 634,845 bytes, 10,749 rows received
04/29/97 10:41:01 TABLE "ACCESSRIGHTS" - 669,242 BYTES, 11,332
ROWS ARCHIVED
04/29/97 10:41:01 --- Starting table "ACCOUNTS"
04/29/97 10:41:02 --- CHKPT: 1,228 bytes, 17 rows received
04/29/97 10:41:02 --- CHKPT: 1,228 bytes, 17 rows received
04/29/97 10:41:04 --- CHKPT: 1,734 bytes, 28 rows received
04/29/97 10:41:11 --- CHKPT: 2,608 bytes, 47 rows received
04/29/97 10:41:11 TABLE "ACCOUNTS" - 2,930 BYTES, 54 ROWS
ARCHIVED
04/29/97 10:41:11 --- Starting table "DBASE"
04/29/97 10:41:12 --- CHKPT: 4,751 bytes, 13 rows received
04/29/97 10:41:13 --- CHKPT: 7,711 bytes, 23 rows received
04/29/97 10:41:14 --- CHKPT: 14,599 bytes, 47 rows received
04/29/97 10:41:14 TABLE "DBASE" - 17,556 BYTES, 57 ROWS ARCHIVED
04/29/97 10:41:14 "JCK" - LOCK RELEASED
04/29/97 10:41:14 DUMP COMPLETED
04/29/97 10:41:14 STATEMENT COMPLETED
04/29/97 10:41:14
04/29/97 10:41:14
04/29/97 10:41:14
04/29/97 10:41:14 RESTORE DATA TABLES JCK),release lock,
file=archive;
04/29/97 10:41:15 --- ARCHIVED AT 04-29-97 10:40:43
04/29/97 10:41:15 --- ARCHIVED FROM ALL AMP DOMAINS
04/29/97 10:41:15 --- UTILITY EVENT NUMBER IN ARCHIVE - 79
04/29/97 10:41:16 UTILITY EVENT NUMBER - 80
04/29/97 10:41:22 STARTING TO RESTORE DATABASE "JCK"
04/29/97 10:41:26 DICTIONARY RESTORE COMPLETED
04/29/97 10:41:29 --- Starting table "ACCESSRIGHTS"
04/29/97 10:41:30 --- CHKPT: 145,204 bytes, 2,450 rows sent
04/29/97 10:41:30 --- CHKPT: 145,204 bytes, 2,450 rows sent
04/29/97 10:41:30 --- CHKPT: 215,414 bytes, 3,640 rows sent
04/29/97 10:41:31 --- CHKPT: 325,449 bytes, 5,505 rows sent
04/29/97 10:41:31 --- CHKPT: 448,582 bytes, 7,592 rows sent
04/29/97 10:41:32 --- CHKPT: 558,440 bytes, 9,454 rows sent
04/29/97 10:41:32 --- CHKPT: 667,059 bytes, 11,295 rows sent
04/29/97 10:41:32 --- Building fallback subtable for index 0.
04/29/97 10:41:32 --- Clearing build flag for index 0.
04/29/97 10:41:33 --- Building index subtable for index 4.
04/29/97 10:41:33 --- Clearing build flag for index 4.
04/29/97 10:41:34 --- Building index subtable for index 8.
04/29/97 10:41:34 --- Clearing build flag for index 8.
04/29/97 10:41:34 "ACCESSRIGHTS" - 669,242 BYTES, 11,332 ROWS
RESTORED
04/29/97 10:41:34 --- Starting table "ACCOUNTS"
04/29/97 10:41:35 --- CHKPT: 1,320 bytes, 19 rows sent
04/29/97 10:41:35 --- CHKPT: 2,010 bytes, 34 rows sent
04/29/97 10:41:35 --- CHKPT: 2,838 bytes, 52 rows sent
04/29/97 10:41:35 --- Building fallback subtable for index 0.
04/29/97 10:41:36 --- Clearing build flag for index 0.
04/29/97 10:41:36 "ACCOUNTS" - 2,930 BYTES, 54 ROWS RESTORED
04/29/97 10:41:59 --- Starting table "DBASE"
04/29/97 10:42:00 --- CHKPT: 5,325 bytes, 15 rows sent

```

```
04/29/97 10:42:00 --- CHKPT: 9,146 bytes, 28 rows sent
04/29/97 10:42:00 --- CHKPT: 15,834 bytes, 51 rows sent
04/29/97 10:42:00 --- CHKPT: 15,834 bytes, 51 rows sent
04/29/97 10:42:00 --- Building fallback subtable for index 0.
04/29/97 10:42:01 --- Clearing build flag for index 0.
04/29/97 10:42:01 --- Building index subtable for index 4.
04/29/97 10:42:02 --- Building fallback subtable for index 4.
04/29/97 10:42:02 --- Clearing build flag for index 4.
04/29/97 10:42:02 "DBASE" - 17,556 BYTES, 57 ROWS RESTORED
04/29/97 10:42:03 "JCK" - LOCK RELEASED
04/29/97 10:42:03
04/29/97 10:42:03 STATEMENT COMPLETED
04/29/97 10:42:03
04/29/97 10:42:03
04/29/97 10:42:03 LOGOFF;
04/29/97 10:42:03 LOGGED OFF 6 SESSIONS
04/29/97 10:42:03 STATEMENT COMPLETED
04/29/97 10:42:03
04/29/97 10:42:03
04/29/97 10:42:03 UTILITY TERMINATED
```

WORKDIR

Purpose

The WORKDIR parameter specifies the name of the working directory.

Syntax

WORKDIR _____ = *directory path* —————▶
WDIR _____
KM01A004

Usage Notes

When you specify a directory with WORKDIR, the restart log, the output log, the error log, and the hard disk archive files are created in and read from this directory. When you do not specify WORKDIR, the current directory is the default working directory.

Delimiters depend on the name of the working directory being specified.

Example

```
WORKDIR=C:\TESTARC  
WORKDIR='C:\TEST ARC'
```

Return Codes and UNIX Signals

This chapter describes these topics:

- [Return Codes](#)
- [UNIX Signals](#)

Return Codes

Teradata ARC provides a return (or condition) code value to the client operating system during termination logic. The return code indicates the final status of a task.

Return code integers increase in value as the events causing them increase in severity, and the presence of a return code indicates that at least one message that set the return code is printed in the output file. Any number of messages associated with a lesser severity might also be printed in the output file. For more information about error messages, see the *Messages* guide.

Table 12 lists the Teradata ARC return codes, their descriptions, and the recommended solution for each.

Table 12: Return Codes

Return Code	Description	Recommended Action
0	A normal termination of the utility.	
4	At least one warning message is in the output listing.	Review the output to determine which warnings and possible consequences are printed.
8	At least one nonfatal error message is in the output listing. A nonfatal error normally indicates that you requested something that could not be done. A nonfatal error does not adversely affect other requests nor does it terminate task execution.	Review the output to determine which errors are listed, and then deal with each error accordingly.
12	A fatal error message is in the output listing. A fatal error terminates execution. You might be able to restart the task.	Review the output to determine which error occurred and its cause.
16	An internal software error message is in the output listing. An internal error terminates execution.	Gather as much information as possible (including the output listing), and contact Teradata customer support.

Table 13 lists the Teradata Database error message numbers, their associated severity levels, and the associated return codes.

The six error messages listed as NORMAL severity are restartable errors that indicate the Teradata Database failed. Teradata ARC waits for the Teradata Database to restart before it automatically resubmits or restarts any affected requests or transactions.

Any other message numbers not shown in this table are treated as FATAL.

Table 13: Database Error Messages

Database Error Message Numbers					Severity Level	Return Code
2825	2828	3120			NORMAL	0
2826	3119	3897				
2123	2838	3803	7486	8269	WARNING	4
2631	2840	3804	8234	8270		
2639	2921	3805	8258	8271		
2641	2971	5419	8261	8272		
2654	2972	5512	8265	8273		
2805	3111	5588	8266	8274		
2835	3598	6933	8267	8295		
2837	3603	7485	8268	8297		
2815	2926	3656	3824	3933	ERROR	8
2830	3523	3658	3853	5310		
2843	3524	3737	3873	8232		
2897	3566	3802	3877	8298		
2920	3613	3807	3916			
2538	2866	8230	8245	8250	FATAL	12
2541	2868	8231	8246	8251		
2644	3596	8242	8247	8252		
2809	8228	8243	8248	8253		
2828	8229	8244	8249	8254		
None					INTERNAL	16

Note: The DBSERROR command line parameter allows the specification of an error severity level with Teradata Database error code 2843. For example, to specify that error 2843 is a warning (severity level 4), use this syntax:

```
ARCMAN DBSERROR=(2843,4)
```

[Table 14](#) lists the error message numbers that are generated on the client side, their associated severity levels, and associated return codes.

Table 14: Client-Generated Error Messages

Client Generated Error Message Number (ARCxxxx)						Severity Level	Return Code
5	703	1025	1244	1800		WARNING	4
6	706	1026	1245	1801			
12	707	1027	1249	1803			
13	708	1029	1250	1804			
18	1001	1213	1251	1805			
20	1003	1214	1254	1900			
22	1004	1217	1256	2202			
101	1005	1218	1257				
103	1006	1223	1263				
104	1010	1225	1266				
106	1011	1226	1267				
107	1012	1228	1269				
111	1014	1232	1272				
113	1015	1235	1401				
118	1016	1239	1402				
120	1022	1240	1403				
203	1023	1241	1404				
225	1024	1242	1501				
108	1206	1409				ERROR	8
119	1218	1410					
715	1227	2100					
802	1248	2102					
1013	1268	2103					
1017	1273	2104					
1030	1406	2107					
1200	1407	2201					
1202	1408						

Table 14: Client-Generated Error Messages (continued)

Client Generated Error Message Number (ARCxxxx)					Severity Level	Return Code
3	207	240	1028	1262	FATAL	12
4	208	241	1201	1264		
7	209	242	1203	1265		
8	210	243	1204	1270		
9	211	244	1207	1271		
10	212	600	1208	1400		
11	213	601	1209	1405		
14	214	700	1210	1500		
15	215	702	1211	1802		
16	216	701	1215	1806		
17	217	704	1216	1901		
19	218	705	1219	2000		
21	219	710	1220	2001		
23	220	711	1221	2002		
24	221	712	1222	2003		
25	222	713	1230	2004		
26	223	714	1231	2101		
100	224	716	1233	2105		
102	226	800	1234	2106		
105	227	801	1236	2200		
109	228	803	1237	2203		
110	229	804	1238	2204		
112	230	805	1243	2205		
114	231	806	1246			
115	232	807	1247			
116	233	808	1252			
117	234	1000	1253			
200	235	1002	1255			
201	236	1018	1258			
202	237	1019	1259			
204	238	1020	1260			
205	239	1021	1261			
1	206				INTERNAL	16

UNIX Signals

UNIX signals are predefined messages sent between two UNIX processes to communicate the occurrence of unexpected external events or exceptions.

A knowledge of UNIX signals is important if you are running Teradata ARC in a UNIX operating system because you cannot use the Teradata ARC UNIX signals in any module or routine that you program for use with Teradata ARC. When one of these signals is received, Teradata ARC terminates with a return code of 16.

Teradata ARC uses these UNIX signals:

- SIGILL (illegal signal)
- SIGINT (interrupt signal)
- SIGSEGV (segmentation violation signal)
- SIGTERM (terminate signal)

Archive/Recovery Control Language

The syntax of Teradata ARC control language is similar to Teradata SQL, that is, each statement is composed of words and delimiters and is terminated by a semicolon. Teradata ARC writes each statement (and a response to each statement) to an output file.

Note: For simplified descriptions of the Teradata ARC control language, experienced Teradata ARC users might want to refer to the Archive/Recovery chapter of the *Teradata Tools and Utilities Command Summary*, which provides only syntax diagrams and a brief description of the syntax variables for each Teradata client utility.

The Teradata ARC control statements perform these types of activities:

- Session statements begin and end utility tasks
- Archive statements perform archive and restore activities, and build indexes
- Recovery statements act on journals, and rollback or rollforward database or table activity
- Data encryption

[Table 15](#) summarizes Teradata ARC statements. For more detail, refer to the sections that follow the table.

Table 15: Summary of Teradata ARC Statements

Statement	Activity	Function
ANALYZE	Archive	Reads an archive tape and displays content information
ARCHIVE	Archive	Archives a copy of database content to a client resident file in the specified format
BUILD	Archive	Builds indexes for data tables and fallback data for fallback tables
CHECKPOINT	Recovery	Marks a journal table for later archive or recovery activities
COPY	Archive	Re-creates copies of archived databases or tables. Also, restores archived files from one system to another.
DELETE DATABASE	Miscellaneous	Deletes a partially restored database
DELETE JOURNAL	Recovery	Deletes a saved or restored journal
ENABLE DATA EXTENSION	Miscellaneous	Enables an extension module
ENABLE ENCRYPTION	Encryption	Enables an encryption extension module

Table 15: Summary of Teradata ARC Statements (continued)

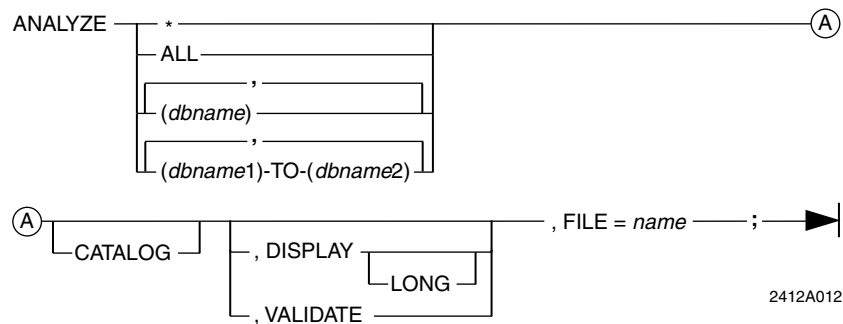
Statement	Activity	Function
LOGDATA	Security	Specifies any additional logon or authentication data required by an extended security mechanism.
LOGGING ONLINE ARCHIVE OFF	Archive	Ends the online archive process.
LOGGING ONLINE ARCHIVE ON	Archive	Begins the online archive process.
LOGMECH	Security	Sets the extended security mechanism used by Teradata ARC to log onto a Teradata Database.
LOGOFF	Session Control	Ends a Teradata session and terminates Teradata ARC
LOGON	Session Control	Begins a Teradata session
QUIT	Session Control	See LOGOFF .
RELEASE LOCK	Miscellaneous	Releases a utility lock on a database or table
RESTORE	Archive	Restores objects from an archive file to specified AMPs
REVALIDATE REFERENCES FOR	Archive	Validates referenced indexes that have been marked inconsistent during restore
ROLLBACK	Recovery	Restores a database or tables to a state that existed before some change
ROLLFORWARD	Recovery	Restores a database or tables to its (their) state following a change
SET QUERY_BAND	Miscellaneous	Provides values for a session to precisely identify the origins of a query.

ANALYZE

Purpose

The **ANALYZE** statement reads an archive tape and displays information about its content.

Syntax



where

Syntax Element	Description
ALL *	Analyzes all databases in the archive file.
CATALOG	Generates/rebuilds the CATALOG table in the CATALOG database. See CATALOG in Chapter 4: “Runtime Parameters”
(dbname)	Name of the database to analyze.
(dbname1) TO (dbname2)	List of alphabetically client databases. The delimiting database names need not identify actual databases. Database DBC is not included as part of the range.
DISPLAY	Displays information about the identified databases.
FILE	Analyzes a file.
LONG	Displays the names of all data tables, journal tables, stored procedures, views, macros and triggers within the specified databases.
name	Name of the archive data set to analyze.
VALIDATE	Reads each archive record for the specified database to check that each block on the file can be read.

Usage Notes

The **ANALYZE** statement provides the following information about the specified databases:

- Time and date the archive operation occurred.
- Whether the archive is of all AMPs, clusters of AMPs, or specific AMPs.
- The name of each database, data table, journal table, stored procedure, view and macro in each database, and the fallback status of the tables (if you specify the LONG option).
- The number of bytes and rows in each table (if you specify either the LONG or VALIDATE option).
- If an archive file contains a selected partition archive of a table, the bounding condition used to select the archived partitions is displayed with an indication as to whether the bounding condition is well-defined.
- Online logging information.
- User-Defined Method (UDM) and User-Defined Type (UDT) information.

It is possible to specify both the DISPLAY and VALIDATE options in a single **ANALYZE** statement. The default is DISPLAY.

Alternate Character Set Support

The **ANALYZE** statement requires a connection to the Teradata Database when object names are specified in internal RDBMS hexadecimal format ('<...>'XN) on the input statement, because Teradata ARC cannot normally accept object names in internal RDBMS format. Although the **ANALYZE** statement does not typically require a connection to the Teradata Database, under this circumstance the object names in internal hexadecimal must be translated by the Teradata Database into the external client format (X'..'). to be used by the ARCMAN program. This connection is done by the **LOGON** statement.

Failure to make a connection to the Teradata Database results in the object name being skipped by the ANALYZE operation, which produces the following warning message:

```
ARC0711 DBS connection required to interpret object name. %s skipped.
```

In addition, when the HEX parameter is executed, HEX display of object names from the **ANALYZE** statement is in the external client format (X'..'). This is because object names in **ANALYZE** statements may contain “hybrid” EBCDIC and kanji EUC, or EBCDIC and kanjiShiftJIS single byte and multibyte characters, which cannot be translated into internal RDBMS format, but can be translated into external client format.

Note: Hybrid object names result when you archive an object created in the KANJIEUC or KANJISJIS character set from a KANJIEBCDIC mainframe client, but the hexadecimal representation of the object being archived is not used. In such a circumstance, the Teradata Database cannot properly translate the character set and produces a hybrid object name composed of single byte characters from the EBCDIC character set and multibyte characters from the creating character set (UNIX or DOS/V). The hybrid object name cannot be understood by the Teradata Database.

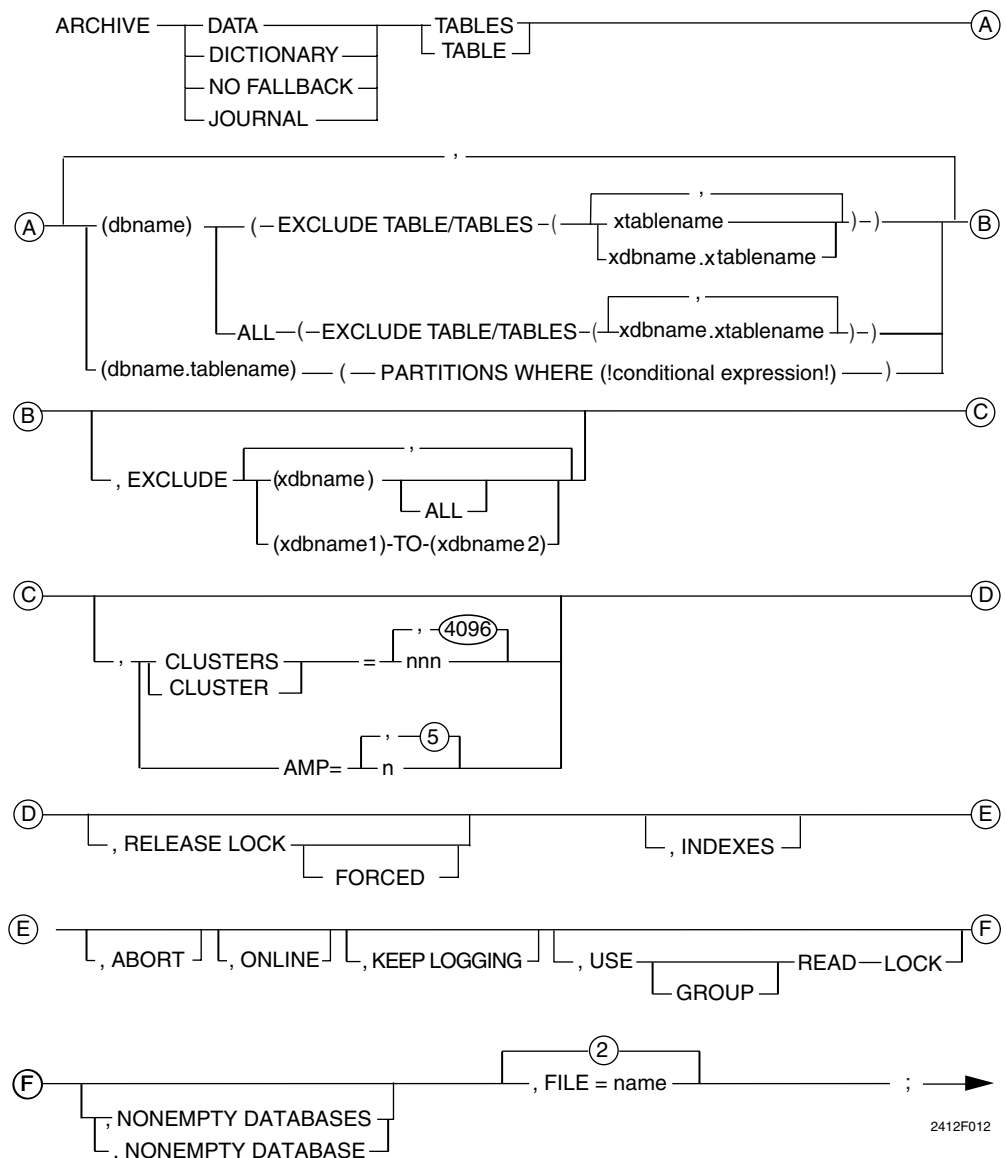
ARCHIVE

Purpose

The **ARCHIVE** statement archives a copy of a database or table to some type of portable media. Use **ARCHIVE** to extract data *from* a Teradata Database. Use the **COPY** or **RESTORE** statement to import data *to* a Teradata Database.

Note: **DUMP** is an alias of **ARCHIVE**.

Syntax



where

Syntax Element	Description
DATA TABLE or DATA TABLES	Archives fallback, non-fallback, or both types of tables from all AMPs or from clusters of AMPs.
DICTIONARY TABLE or DICTIONARY TABLES	Archives only the dictionary rows of a table. A dictionary archive of a database includes all table, view, macro and trigger definitions, and dictionary entries for stored procedures. This option includes the table definition for the specified table in the archive only if you archive a specific table.
NO FALLBACK TABLE or NO FALLBACK TABLES	Archives non-fallback table from specific AMPs to complete a previous all-AMPs or cluster archive taken with processors offline.
JOURNAL TABLE or JOURNAL TABLES	Archives dictionary rows for journal table and the saved subtable of journal table.
<i>(dbname)</i>	Name of the database from which tables are archived. This causes all tables of the specified type in the database to be archived.
ALL	Archives all tables from the named database and its descendants, in alphabetical order.
<i>(dbname.tname)</i>	Name of a table within the named database to archive. All tables must be distinct within the list. Duplicate tables are ignored.
EXCLUDE	Prevents tables in the listed database from being archived. Note that individual tables cannot be excluded using this option. Individual tables can be excluded using the EXCLUDE TABLE(S) option after each database. See more in EXCLUDE TABLE, below.
<i>(xdbname)</i>	Name of a database to exclude.
ALL	Excludes the named database and its descendants.
<i>(xdbname1) TO (xdbname2)</i>	Alphabetical list of client databases to exclude. The delimiting database names need not identify actual databases. Database DBC is not included within a range.
EXCLUDE TABLE or EXCLUDE TABLES	Prevents individual tables in the listed database from being archived.
<i>(tablename)</i>	Name of an individual table in the designated database to exclude. Multiple tables are separated by commas. This form (without a database name prefix) is used only when ALL has not been designated
<i>(xdbname.tablename)</i>	List of fully-qualified tables (i.e., prefixed by database name) to exclude. This form is used only when ALL has been specified with the EXCLUDE TABLES keywords.

Syntax Element	Description
PARTITIONS WHERE	Specifies the conditional expression for selecting partitions. If the condition selects a partial partition, the entire partition is archived.
(!conditional expression!)	Conditional expression for specifying selected partitions.
CLUSTERS = <i>nnn</i> or CLUSTER = <i>nnn</i>	Specifies AMP clusters to archive, where <i>nnn</i> is the number of the clusters, up to 4096 clusters.
AMP = <i>n</i>	Specifies the AMP (or a list of AMPs) to archive for Teradata Database, where <i>n</i> is the number of the AMP, up to five AMPs.
RELEASE LOCK	Releases utility locks on the identified objects when the archive operation completes successfully. Warning: Releasing a HUT lock while another Teradata ARC job is running could result in data corruption and/or unexpected errors from ARCMAN or the Teradata Database. Refer to “RELEASE LOCK Keyword” on page 143 for more information.
FORCED	When specified, FORCED will instruct ARC to attempt to release any placed locks in the event that the ARCHIVE statement fails. Note: The HUT lock is not guaranteed to be released in all cases. Specifically, the following cases will result in a leftover lock: <ul style="list-style-type: none"> • If ARC is forcibly aborted by the user or operating system. • If communication to the Teradata Database is lost and can not be reestablished. • If an internal failure occurs in ARC, such that program control can not proceed to, or complete, the release lock step.
INDEXES	Archives data blocks used for secondary indexing. This option is not available for archiving selected partitions of PPI tables, although it is allowed for a full-table archive of a PPI table.
ABORT	Aborts an all-AMPs or cluster operation if an AMP goes offline during the archive of non-fallback tables or single image journal tables.
ONLINE	Initiates an online archive on a table or database.
KEEP LOGGING	Overrides the automatic termination of an online archive that was initiated with the ONLINE option.

Syntax Element	Description
USE READ LOCK or USE GROUP READ LOCK	<p>Applies a read or group HUT lock on the entity being archived. This option is available only when archiving data tables.</p> <p>Specify the GROUP parameter only when you archive all AMPs or cluster AMPs. The GROUP keyword is rejected in a specific AMP archive operation.</p> <p>If you specify the GROUP parameter during an archive, you will be unable to use the archive if you are restoring to a system that has a different hash function or configuration than the source system. Therefore, perform an archive with <i>no</i> GROUP parameter <i>prior</i> to a major release upgrade or re-configuration.</p> <p>To use the GROUP READ LOCK option, after-image journaling must be enabled for the tables.</p>
NONEMPTY DATABASES or NONEMPTY DATABASE	<p>Excludes any empty users or databases from the ARCHIVE operation. This option works with all types of archives. If you archive a journal, databases without a journal are excluded.</p>
FILE	<p>Names an output archive file.</p> <p>You can specify this option twice in the same ARCHIVE statement. If you specify the option twice, Teradata ARC generates two identical archive files concurrently.</p> <p>Note: For Windows NT/2000/XP, only one FILE is supported.</p> <p>Warning: If more than one ARCHIVE statement is specified in a script, the value for the FILE parameter is different for each statement. Otherwise, the file created by the second ARCHIVE statement will overwrite the file created by the first ARCHIVE statement.</p>
<i>name</i>	Name of the output archive file.

Access Privileges

To archive a database or a table in a database, specify one of the following for the user name in the **LOGON** statement:

- The ARCHIVE privilege on the database or table to archive
- Ownership of the database or table

To archive database DBC, the user name must either be “User DBC” or be granted the ARCHIVE and SELECT privileges by user DBC.

To archive a journal table, the user name must have the ARCHIVE privilege on one of the following:

- The journal table itself
- The database that contains the journal table

Archiving Process

Database SYSUDTLIB is linked with database DBC and is only archived if DBC is archived. SYSUDTLIB cannot be specified as an individual object in an **ARCHIVE** statement.

If database DBC is involved in an archive operation, it is always archived first, followed by database SYSUDTLIB. If additional databases are being archived, they will follow SYSUDTLIB in alphabetical order.

By default, Teradata ARC counts output sectors and output rows. The row count is the number of primary data rows archived when you specify the DATA or NO FALLBACK option. Both counts are included in the output listing.

Archives have these limitations:

- Journal and data (or dictionary) tables cannot be stored on the same archive file.
- When you archive a journal table, the saved subtable portion of the journal is archived. If you perform two archives without doing a CHECKPOINT WITH SAVE operation in between, the two archive files are identical.
- If you create an archive when the saved subtable is empty, the archive file contains only control information used by Teradata ARC.

Archive Messages

During an archive of a Teradata Database, Teradata ARC issues messages in the format shown below as it archives tables, views, macros, stored procedures, triggers, and permanent journals. The format of the byte and row counts agree with the format of these counts in the restore or copy output.

```
FUNCTION "UDFname" - n,nnn,nnn BYTES, n,nnn,nnn ROWS ARCHIVED
JOURNAL "Journalname" - n,nnn,nnn BYTES ARCHIVED
MACRO "macroname" - ARCHIVED
METHOD "methodname" - ARCHIVED
METHOD "methodname" - n,nnn,nnn BYTES, n,nnn,nnn ROWS ARCHIVED
PROCEDURE "procedurename" - n,nnn,nnn BYTES, n,nnn,nnn ROWS ARCHIVED
TABLE "tablename" - n,nnn,nnn BYTES, n,nnn,nnn ROWS ARCHIVED
TRIGGER "triggername" - ARCHIVED
TYPE "UDTname" - ARCHIVED
VIEW "viewname" - ARCHIVED
```

When you archive the objects, Teradata ARC issues this message:

```
DUMP COMPLETED
```

At the start of each database archive operation or after a system restart, the output listing shows all offline AMPs.

If an archive file contains a selected partition archive of a table, a message reports the bounding condition that was used to select the archived partitions and whether the bounding condition is well-defined.

If online logging is initiated for one or more tables during an archive, online logging information is displayed for each of the tables in the output listing.

Archive Results

After each archive operation, review the listing to determine whether any non-fallback tables were archived while AMPs were offline. If any AMPs were offline during the archive, you should create a specific AMPs archive for those tables. When the offline AMPs are brought back online, use this archive to complete the operation.

Teradata ARC does not completely archive data tables that are in the process of being restored or data tables that are involved in a load operation (by FastLoad or MultiLoad). If tables are being restored or loaded, Teradata ARC archives only enough data to allow tables to be restored empty and issues a message.

Archive Limitation

Tables with *unresolved* referential integrity constraints cannot be archived. An unresolved constraint occurs when a **CREATE TABLE** (child) statement references a table (parent) that does not exist. Use the SQL command **CREATE TABLE** to create the parent (that is, referenced) table and resolve these constraints. For more information, refer to the *Introduction to Teradata Warehouse*.

Using Keywords with ARCHIVE

The **ARCHIVE** statement is affected by the following keywords. For information about keywords that are specific to archiving selected partitions in a PPI table, see [“Archiving Selected Partitions of PPI Tables” on page 146](#).

AMP Keyword

With Teradata Database, use the *AMP=n* option to specify up to five AMPs. If a specified processor is not online when the archive is for a particular database or table, Teradata ARC does not archive any data associated with that AMP.

This option archives individual processors and is useful for following up an all-AMPs or cluster archive when the AMPs were offline. Use this option only with the **NO FALLBACK TABLES** or **JOURNAL TABLES** option. Specify the option for (database DBC) **ALL** to automatically exclude database DBC.

CLUSTER Keyword

If you use the **CLUSTER** keyword, maintain a dictionary archive for the same databases, stored procedures, or tables. Cluster archiving improves archive and recovery performance of very large tables and simplifies specific AMP restoring of non-fallback tables.

A complete cluster archive can restore all databases, stored procedures, and tables if it includes:

- A dictionary archive of the databases, tables, or stored procedures
- A set of cluster archives
- A set of specific AMP archives (if you archive non-fallback tables and AMPs were offline during the cluster archive) so all data is contained in archive files

Create a dictionary archive whenever the tables in the cluster archive might be redefined.

Although it is necessary to generate a dictionary archive only once if you do not redefine tables, make a dictionary archive regularly in case an existing dictionary archive is damaged.

If you create a cluster archive over several days *and* if you redefine a table in that archive, create an all-AMPs archive of the changed tables. If you do not do this, the table structure may differ between archive files.

The process when archiving journal tables is to archive the saved subtable created by a previous checkpoint. Consequently, it is not necessary to use locks on the data tables that are using the journal table. The Teradata Database places an internal lock on the saved journal subtable to prevent a concurrent deletion of the subtable.

The following rules govern the CLUSTER keyword:

- The option is allowed only when archiving data tables.
- A dictionary archive must be performed before a cluster archive.
- If any AMPs in the clusters are offline, archive non-fallback tables from those AMPs when they are brought back online.
- Utility locks are placed and released at the cluster level.

The dictionary archive contains data stored in the dictionary tables of the Teradata Database for the tables, views, macros, stored procedures, and triggers. Do the following to ensure that the table data in the cluster archive matches the dictionary definition in the dictionary archive:

- Archive clusters on tables with definitions that never change.
- Create the dictionary archive immediately prior to the cluster archives, without releasing Teradata ARC locks. Then initiate a set of cluster archive requests covering all AMPs.

You cannot archive database DBC in cluster archive operations. You can, however, specify DBC (ALL). In this case, database DBC is excluded automatically by Teradata ARC.

RELEASE LOCK Keyword

Warning: Releasing a HUT lock while another Teradata ARC job is running could result in data corruption and/or unexpected errors from ARCMAN or the Teradata Database.

This can occur if a HUT lock is explicitly released during an ARC job, or if multiple jobs are run concurrently on the same object with the same Teradata user, with each job specifying the RELEASE LOCK option. For example, a database-level archive, with one or more tables excluded, is run concurrently with a table-level archive of the excluded tables. In this situation, the RELEASE LOCK for the database-level job will release the table-level locks if the job completes first.

Use different Teradata users for each archive job when running concurrent archives against the same objects. This ensures that locks placed by one job are not released by another.

For the purposes of locking, a set of cluster archive jobs, with each job archiving different clusters, counts as only a single job running against the object. If all concurrent jobs running on an object archive different sets of clusters, it is acceptable to use the same Teradata user for each of the cluster archive jobs.

INDEXES Keyword

Specifying the INDEXES keyword increases the time and media required for an archive. This option does not apply to journal table archives. The option is ignored for cluster and specific AMP archives.

The INDEXES keyword usually causes Teradata ARC to archive unique and non-unique secondary indexes on all data tables. However, if an AMP is offline, Teradata ARC only archives unique secondary indexes on fallback tables. Therefore, for the INDEXES keyword to be most effective, do not use it when AMPs are offline.

If the archive applies a group read HUT lock, Teradata ARC writes no indexes to the archive file.

ABORT Keyword

For cluster archives, Teradata ARC only considers AMPs within the specified clusters. The ABORT keyword does not affect specific AMP archives.

GROUP Keyword

To avoid the full read HUT lock that Teradata ARC normally places, specify USE GROUP READ LOCK for a rolling HUT lock. (It is also possible to avoid the full read HUT lock by initiating an online archive using the ONLINE option during an ALL AMPS archive, or by using **LOGGING ONLINE ARCHIVE ON.**)

If you specify the GROUP keyword, the Teradata Database:

- 1 Applies an access HUT lock on the whole table.
- 2 Applies a read HUT lock sequentially on each group of rows in the table as it archives them, then:
 - Releases that read HUT lock.
 - Applies another read HUT lock on the next group of rows to be archived.
- 3 After the last group of rows is archived, releases the last group read HUT lock and the access HUT lock on the table.

Using the GROUP keyword locks out updates to a table for a much shorter time than an archive operation without the option. The size of the locked group is approximately 64,000 bytes.

A transaction can update a table while the table is being archived under a group read HUT lock. If an update occurs, the archive might contain some data rows changed by the transaction and other data rows that were not changed. In this case, the archived copy of the table is complete only when you use it in conjunction with the after image journal created by the transaction.

Use a group read HUT lock only for tables with an after image journal. This means that online backups of database-level objects also require every table in the database have after-journaling defined. If a table is excluded as part of a database-level GROUP READ LOCK archive, it must still have an after-journal table defined so ARC can accept the GROUP READ LOCK option.

If you specify a group read HUT lock for tables without an after image journal, the Teradata Database does not archive the table, and it prints an error message.

If you do not specify the GROUP keyword and the archive operation references a database, the Teradata Database locks the database. This HUT lock is released after all data tables for the database have been archived.

If the **ARCHIVE** statement specifies the archive of individual data tables, the Teradata Database locks each table before it is archived and releases the HUT lock only when the archive of the table is complete. HUT locks are released only if you specify the RELEASE LOCK keywords.

ONLINE Keyword

The ONLINE keyword initiates an online archive on a table or database. During an online archive, the system creates and maintains a log for the specified table or a separate log for each table in the specified database. All changes to a table are captured in its log. Each log of changed rows is then archived as part of the archive process. When the archive process completes, the online archive is terminated and the change logs are deleted.

When a table is restored, the table's change log is restored also. The changed rows in the log are rolled back to the point that existed prior to the start of the archive process. The table is then rolled back to that same point. All tables within the same transaction in the online archive are restored to the same point in the transaction.

KEEP LOGGING Keyword

The KEEP LOGGING keyword overrides automatic termination of:

- an ALL AMPs online archive that was initiated with the ONLINE option
- a cluster online archive that was initiated with **LOGGING ONLINE ARCHIVE ON**

When KEEP LOGGING is used, the online archive process continues after the archive process completes. Changed rows continue to be logged into the change log. Use this keyword with caution: online logging continues until it is terminated manually with **LOGGING ONLINE ARCHIVE OFF**. If online logging is not terminated, there is a possibility that the logs could fill all available perm space.

NONEMPTY DATABASE Keywords

Teradata ARC writes no information to tape about empty objects. If there are many empty users on a system, using these keywords can appreciably reduce the archive time.

The NONEMPTY keyword is only meaningful when used in conjunction with the ALL option. Using NONEMPTY, without ALL for at least one object, produces an ARC0225 warning message. In this case, the NONEMPTY option is ignored and execution continues.

Teradata ARC uses the *ARC_NonEmpty_List* macro to determine which databases to archive. If this macro was not installed or if you do not have the EXECUTE privilege, Teradata ARC still archives all objects. Thus, if this macro is not installed, the time savings is not realized.

Archiving Selected Partitions of PPI Tables

It is possible to perform an all-AMPs archive on one or more partitions of a table rather than performing a full-table backup and restore. This feature is limited to all-AMP archives. Dictionary, cluster, and journal archives are not supported.

Use selected partitions to archive only a subset of data and avoid archiving data that has already been backed up. (Minimizing the size of the archive can improve performance.)

Before using this feature, be sure to understand [“Potential Data Risks When Archiving/Restoring Selected Partitions” on page 33](#).

For procedures and script examples of selecting partitions, see [“Archiving Selected Partitions of PPI Tables” on page 30](#).

Restrictions on Archiving Selected Partitions

These restrictions apply to archiving selected partitions of a PPI table:

- Cluster, dictionary, and journal archives are not supported.
- It is recommended that tables containing large objects (BLOB and CLOB columns) not be archived with selected partitions (using PARTITIONS WHERE) because a number of manual steps are needed to restore the data. Instead of using selected partitions, use a full-table archive and restore for tables that contain both PPIs and LOBs.

For additional information, see [“Potential Data Risks When Archiving/Restoring Selected Partitions” on page 33](#) and [“Considerations Before Restoring Data” on page 40](#).

Keywords for Archiving Selected Partitions

To perform an archive of selected partitions, specify a conditional expression in the PARTITIONS WHERE option in an ARCHIVE script. This conditional expression should only reference the column(s) that determine the partitioning for the table being archived.

PARTITIONS WHERE Keyword

Use the PARTITIONS WHERE option to specify the conditional expression, which contains a definition of the rows that you want to archive. To be effective, limit this expression to the columns that determine the partitioning for the table you want to archive.

These restrictions apply to the use of PARTITIONS WHERE:

- The object is an individual table (not a database).
- The source table has a PARTITIONS BY expression defined.
- The archive is an all-AMP archive (not a dictionary, cluster, or journal archive).
- The INDEXES option is not used.
- If the table belongs to a database that is specified in the **ARCHIVE** statement, the table is excluded from the database-level object (with EXCLUDE TABLES) and is individually specified.
- Any name specified in the conditional expression is within the table being specified. (Using table aliases and references to databases, tables, or views that are not specified within the target table result in an error.) It is recommended that the only referenced

columns in the PARTITIONS WHERE condition be the partitioning columns or system-derived column PARTITION of the table. References to other columns do not contribute to partition elimination, and might accidentally qualify more partitions than intended.

Examples of ARCHIVE Keywords

This example executes an archive of all the rows in the TransactionHistory table in the SYSDBA database for the month of July 2002:

```
ARCHIVE DATA TABLES
(SYSDBA.TransactionHistory)
(PARTITIONS WHERE
(! TransactionDate BETWEEN DATE '2002-07-01' AND DATE '2002-07-31' !)) )
,
RELEASE LOCK,
FILE=ARCHIVE;
```

Changes Allowed to a PPI Table

The following changes can be made to a PPI table without affecting or preventing the restoration of data from an archive of selected partitions:

- Table-level options that are unrelated to semantic integrity, such as FALLBACK protection, journaling attributes, free space percentage, block size, and others.
- Use of the MODIFY PRIMARY INDEX option, which changes the partitioning expression.
- Changes to table-level CHECK CONSTRAINTS.
- Changes that add, drop, or modify a CHECK CONSTRAINT on a column.
- Secondary indexes may be added or dropped.

Other changes to a PPI table are more significant in that they affect the *DBC.TVM.UtilVersion*. You cannot restore archives of selected partitions if the archive *UtilVersion* does not match the table *UtilVersion*. *DBC.TVM.UtilVersion* is initially set to 1. ALTER TABLE increases the value of *DBC.TVM.UtilVersion* to match the table *DBC.TVM.Version* whenever the following significant changes occur to the referential integrity of a child table:

- Modification of the columns of a primary index
- Addition or deletion of columns
- Modification of the definition of an existing column
- Addition, deletion, or modification of the referential integrity among tables

When *DBC.TVM.UtilVersion* is updated for a table, previous archives are *invalid* for future restores or copies of selected partitions to that table, but a full table restore or copy is still valid. For more information, see the *Data Dictionary*.

Table-Level Exclude Option with ARCHIVE

The table-level exclude object option is allowed in an ARCHIVE statement (and in COPY and RESTORE statements). This option is only accepted in a database-level object in a DATA TABLES of all-AMPs or cluster operations. A database-level object that has one or more excluded tables is a partial database.

An archive of a partial database contains the dictionary information and the table header row of the excluded table. However, the actual data rows are not archived. If a partial database archive is restored, no data rows are restored for the excluded tables.

If a table is excluded, Teradata ARC restores the dictionary information and the table header row, but leaves the table in restore state. This prevents the table from being accessed by another application before the table level restore is performed. A table-level restore for the excluded tables is expected to follow the partial database restore to fully restore a partial database.

To exclude a table, run a **BUILD** statement for the excluded tables. The excluded tables become accessible and are empty; they can then be dropped.

If ALL keyword is specified after the object name, then only fully qualified table names in the form of databasename.tablename are accepted in the list of EXCLUDE TABLES. Do not use EXCLUDE TABLES with the following options: - table level object: (db.tb) - DICTIONARY, JOURNAL, NO FALLBACK - AMP=. ARC0215 will be issued if above condition is detected.

Caution: During a full database-level restore of an archive with excluded tables, the data dictionaries and the table headers of *all* tables, including excluded tables, are replaced. As a result, *all* of the existing rows in the excluded tables are deleted. To leave an existing excluded table intact during a **RESTORE** or **COPY** operation, use the EXCLUDE TABLES option on the **RESTORE** or **COPY** statement.

You can restore individual tables from a database-level archive with excluded tables. In the **RESTORE** statement, individually specify all the tables you want to restore, except the excluded tables. By omitting the excluded tables, you preserve the data dictionaries and table headers of the excluded tables. That way you can restore the database from the archive without altering the excluded tables.

However, you cannot name macros, views, stored procedures, triggers, or UDFs as objects in the **RESTORE** statement. Consequently, if you create an archive with excluded tables and you want to preserve the excluded tables, you cannot recover those objects from the archive.

Table-Level Exclude Errors

The following error messages can occur when you use the table-level exclude object option in an **ARCHIVE** statement.

ARC0106: "User excluded table(s) (%s) does/do not belong to database %s"

One or more tables specified in the EXCLUDE TABLES object option do not belong to the database object.

This is only a warning. You do not need to take any action; the archival will continue.

ARC0107: "%s is a %s. Not excluded."

Views, macros, stored procedures, and triggers are not allowed in the EXCLUDE TABLE list. Only tables are allowed in the EXCLUDE TABLE list. Teradata ARC ignores this entry and continues the operation.

Remove the entry from the EXCLUDE TABLE list.

ARC0108: "Invalid table name in EXCLUDE TABLE list: %s.%s"

This error occurs when a table specified in EXCLUDE TABLE list does not exist in the database. Correct the table name and resubmit the job.

ARC0109: "One or more tables in EXCLUDE TABLE list for (%s) ALL is/are not valid"

This error occurs when validation of one or more tables in the EXCLUDE TABLE list failed. The invalid table names displayed prior to this error. Correct the database/table name and resubmit the job.

ARC0110: "%s is not parent database of %s in EXCLUDE TABLE list"

This error occurs when a table specified in EXCLUDE TABLE list does not belong to any of the specified parent's child databases. Correct the database/table name and resubmit the job.

ARC0710: "Duplicate object name has been specified or the table level object belongs to one of database level objects in the list: %s"

The severity of ARC0710 has been changed to FATAL (12) from WARNING (4).

Do not specify excluded tables in a statement that also contains the database excluding them (unless you use the PARTITIONS WHERE option to archive selected partitions of PPI tables, in which excluded tables are allowed). For example:

```
ARCHIVE DATA TABLES (db) (EXCLUDE TABLES (a)), (db.a), RELEASE LOCK,
FILE=ARCHIVE; /* ARC0710 */
```

This fatal error resulted from one of the following:

- the specified object is a duplicate of a previous object in the list
- the specified object is a table level object that belongs to one of the database-level objects in the list
- the ALL option has been specified and one of the child databases is a duplicate of an object explicitly specified in the list

Remove one of the duplicate names or the table level object and resubmit the script.

ARC1242 "Empty Table restored/copied: %s"

When restoring a table excluded by a user, this warning will be displayed. It normally displays during archiving for the tables that have been excluded. No data rows were restored or copied. Only the dictionary definition and the table header row were restored or copied.

This warning indicates that a table-level restore/copy must follow in order to fully restore the database.

The table is currently in restore state. You can either:

- Complete the partial restore of the database by running a table level restore/copy of the table, or
- Access the table by running an explicit build. This results in an empty table.

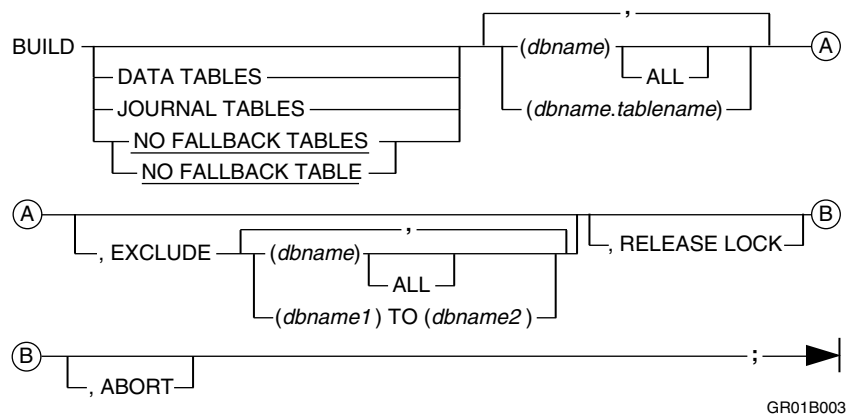
BUILD

Purpose

The **BUILD** statement generates:

- Indexes for fallback and non-fallback tables
- Fallback rows for fallback tables
- Journal tables by sorting the change images

Syntax



where

Syntax Element	Description
DATA TABLES JOURNAL TABLES NO FALLBACK TABLES or NO FALLBACK TABLE	Keywords specifying the type of table to build. The default is NO FALLBACK TABLE. Specify DATA TABLES when building fallback, nonfallback, or both types of tables from all AMPs. This option normally follows the restore of a cluster archive. Specify NO FALLBACK TABLE only when building indexes for non-fallback tables.
(dbname)	Name of a database.
ALL	Keyword to indicate the operation affects the named database and its descendants.
(dbname.tablename)	Name of a non-fallback table for which indexes are built.
EXCLUDE	Keyword to prevent indexes from being built in specified databases.

Syntax Element	Description
<i>(dbname)</i>	Name of database excluded from the build operation.
ALL	Keyword to indicate the operation affects the named database and its descendants.
<i>(dbname1) TO (dbname2)</i>	List of alphabetically client sorted databases excluded from the build operation. The delimiting database names need not identify actual databases. Database DBC is not included as part of a range.
RELEASE LOCK	Keywords to release utility locks on the specified objects when the build operation completes successfully.
ABORT	Keyword to abort the build operation if an AMP is offline <i>and</i> if the object being built is either a nonfallback table or is a database that contains a nonfallback table.

Access Privileges

The user name specified in the **LOGON** statement must have one of the following:

- RESTORE privileges on the specified database or table
- Ownership of the database or table

Usage Notes

Teradata ARC processes databases, tables, and stored procedures in alphabetical order. Because database DBC does not contain non-fallback tables and cannot be restored by clusters, Teradata ARC does not try to build indexes or fallback data for any table in database DBC.

A build operation creates unique and non-unique secondary indexes for a specified table. The build operation also generates the fallback copy of fallback tables if you specify the DATA TABLES keywords.

Typically, this statement is used after one of the following:

- Specific AMP restore of non-fallback tables
- Cluster archive restoring
- Series of all-AMPs restores that specified the NO BUILD option (usually a restore to a changed hardware configuration)

Note: To speed the recovery of all AMPs, specify the NO BUILD option of the **RESTORE** statement to prevent non-fallback tables from being built after the restore is complete.

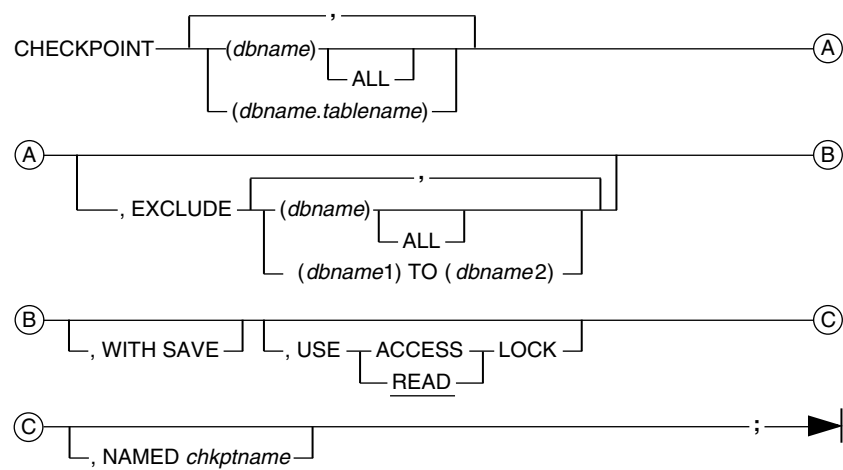
If an AMP is offline during a build operation, all unique secondary indexes of non-fallback tables are left invalidated. As a result, requests may run more slowly when secondary indexes are used for search and selection. Insert and update operations cannot occur.

CHECKPOINT

Purpose

The **CHECKPOINT** statement places a bookmark in a permanent journal or prepares a permanent journal for archiving.

Syntax



GR01B004

where

Syntax Element	Description
(dbname)	Name of the database with the journal table to checkpoint.
ALL	Indicates that the operation affects the journals in the named database and all the descendants.
(dbname.tablename)	Name of a specific journal table to checkpoint. All tables should be distinct within the list. Duplicate tables are ignored.
EXCLUDE	Prevents journal tables from being check-pointed.
(dbname)	Name of database excluded from the checkpoint operation.
ALL	Excludes the named database and its descendants.
(dbname1) TO (dbname2)	Alphabetic list of client databases. The delimiting database names need not identify actual databases. Database DBC is not included as part of a range.

Syntax Element	Description
WITH SAVE	Logically moves the contents of the active subtable of the identified journals to the saved subtable.
USE ACCESS LOCK or USE READ LOCK	Applies an access or read lock on data tables that contribute journal rows to the journal tables on which Teradata ARC is setting checkpoints. USE READ LOCK is the default.
NAMED	References the entry in database recovery activities.
<i>chkptname</i>	Name of the database recovery event.

Access Privileges

The user name specified in the **LOGON** statement must have one of the following:

- CHECKPOINT privileges on the journal table or on the database that contains the journal table
- Ownership of the database that contains the journal table

Usage Notes

Specifying CHECKPOINT with the SAVE keyword prepares a journal table for archiving. Teradata ARC writes a checkpoint row to the active journal table. This subtable is then appended to any existing saved table of the named journal. You can then archive this saved portion by using the **ARCHIVE JOURNAL TABLES** statement.

The Teradata Database returns an event number that Teradata ARC assigns to the checkpoint entry.

WITH SAVE Keywords

Only the saved subtable can be archived.

If a saved subtable for any identified journals exists, Teradata ARC appends the active subtable to it. Teradata ARC writes the checkpoint record to the active subtable and then moves the active subtable to the saved subtable.

Without this option, Teradata ARC leaves the active subtable as is after writing a checkpoint record.

USE LOCK Keywords

USE LOCK specifies the lock level Teradata ARC applies to data tables that contribute rows to the journal tables on which Teradata ARC is setting checkpoints.

With USE LOCK, request the specified lock level on the data tables for a journal in the object list. When Teradata ARC obtains the requested lock, it writes the checkpoint and then releases the lock. Teradata ARC then requests locks for data tables for the next journal table in the object list. Teradata ARC places journal tables in the object list in alphabetical order by database name.

If you specify an ACCESS LOCK, you can make updates to data tables while you are generating the checkpoint. If you take a checkpoint with an ACCESS LOCK, any transaction that has written images to the journal and has not been committed is logically considered to have started after the checkpoint was taken.

When specifying an ACCESS LOCK, you must also specify the WITH SAVE option.

If you specify a READ LOCK, Teradata ARC suspends all updates to the tables until it writes the checkpoint record. With READ LOCK, you can identify the transactions included in the archive.

READ LOCK is the default.

NAMED Keyword

If the *chkptname* duplicates an existing entry in a journal, you can qualify the entry by the system assigned event number.

If you do not use *chkptname*, reference the entry by its event number.

Alpha characters in a checkpoint name are not case sensitive. For example, *chkptname* ABC is equal to *chkptname* abc.

COPY

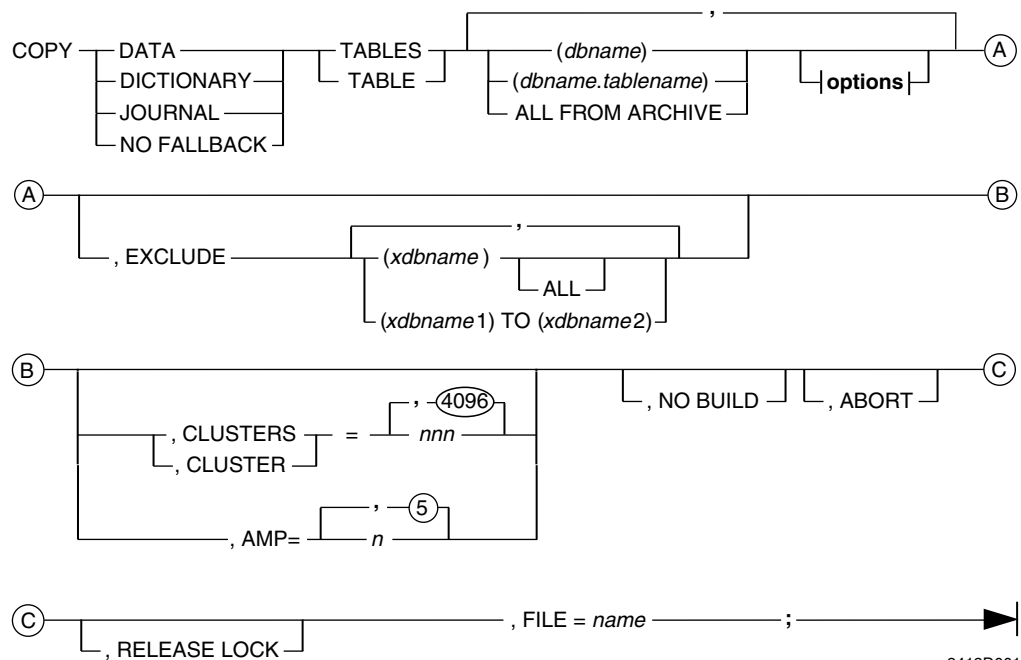
Purpose

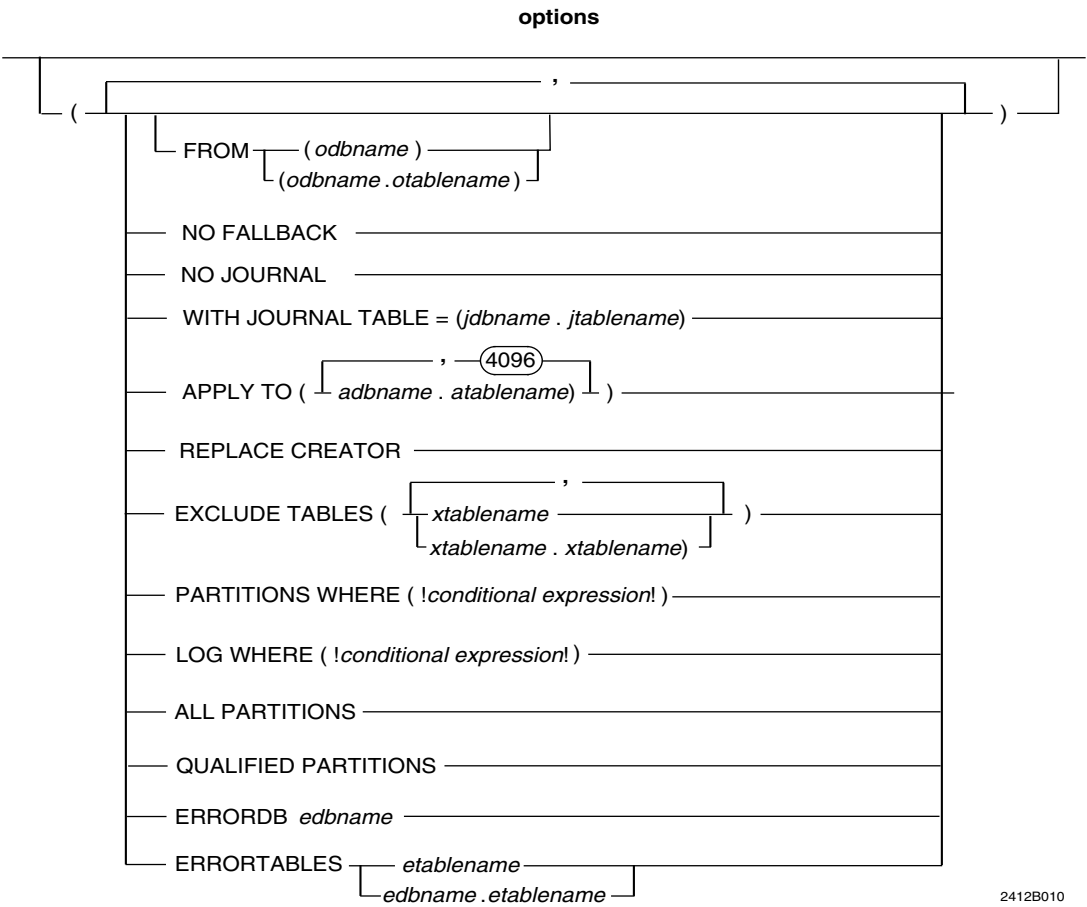
The **COPY** statement copies a database or table from an archived file to the same or different Teradata Database from which it was archived.

Use **COPY** to move data from an archived file back to the Teradata Database. While the target database must exist, COPY creates a new table if it does not already exist on the target database. If a COPY of selected partitions is requested, the target table must exist on the target Teradata Database.

Note: When using COPY, the target database must exist on the target Teradata Database.

Syntax





2412B010

where

Syntax Element	Description
DATA TABLE or DATA TABLES	Copies data tables.
DICTIONARY TABLE or DICTIONARY TABLES	Copies dictionary tables.
JOURNAL TABLE or JOURNAL TABLES	Copies journal tables.
NO FALLBACK TABLE or NO FALLBACK TABLES	Copies non-fallback tables to specific AMPs.
ALL FROM ARCHIVE	Copies all databases/tables in the given archive file.
EXCLUDE	Prevents objects in specified databases from being copied.
(dbname)	Name of database to exclude from copy.
ALL	Excludes the named database and all descendants.

Syntax Element	Description
<i>(xdbname1) TO (xdbname2)</i>	Alphabetical list of client databases to exclude from the copy. The delimiting database names need not identify actual databases. Database DBC is not part of a range.
<i>(dbname) or (dbname.tablename)</i>	Name of the target object. Teradata ARC replaces the target object with the object from the archive.
FROM	Names the object in the archive different from the target object.
NO FALLBACK	Copies fallback tables into non-fallback tables. If the archived table is already non-fallback, this option has no effect.
REPLACE CREATOR	Replaces the LastAlterUID, creator name, and Creator ID of the tables in the target database with the user ID and the current user name, i.e., the user name specified in the LOGON command.
NO JOURNAL	Copies all tables with journaling disabled, whether journaled in the archive or not.
WITH JOURNAL TABLE	Specifies that a copied database has journaling for the specified database and journal table.
APPLY TO	Identifies the tables in the target system where the change images apply.
NO BUILD	Prevents the Fallback and Secondary Index rows from being created. If NO BUILD is requested when restoring database DBC, the request is ignored. If the NO BUILD keywords are used during a RESTORE statement, a separate BUILD statement must be run for all databases and/or tables that were restored. The tables will not be accessible until a BUILD statement is run.
ABORT	Aborts the copy operation if an AMP to which a non-fallback or journal table to restore is offline. This option affects only an all-AMPs copy.
RELEASE LOCK	Releases client utility locks when the copy operation completes.
FILE	Copies a file.
<i>name</i>	Name of the file that contains the archive to copy.
EXCLUDE TABLE or EXCLUDE TABLES	Prevents individual tables in the listed database from being copied.
<i>(xtablename)</i>	Name of an individual table in the designated database to exclude. Multiple tables are separated by commas.
<i>(xdbname.xtablename)</i>	List of fully qualified tables (i.e., prefixed by database name) to exclude.
PARTITIONS WHERE	Specifies the conditional expression for partition-level operations.
<i>(!conditional expression!)</i>	The conditional expression for specifying selected partitions.

Syntax Element	Description
LOG WHERE	Specifies the rows that are logged to an error table for manual insertion and deletion.
(!conditional expression!)	The conditional expression for specifying rows to log to the error table when copying selected partitions.
ALL PARTITIONS	Restores all archived partitions for an archived PPI object. See “ALL PARTITIONS Keyword” on page 166 for conditions that must be met.
QUALIFIED PARTITIONS	Copies the same partitions specified in a previous selected-partition copy.
ERRORDB and ERRORTABLES	Specifies the location of the error log for partition-level operations.

Access Privileges

You must have RESTORE access privileges for the target database or table to use the **COPY** statement. The target database must also exist before you make the copy.

If you copy a single table that did not exist in the destination database before the copy, you must have CREATE TABLE and RESTORE DATABASE access rights for the target database.

Usage Notes

Do not use database DBC as the target database name in a **COPY** statement. You can use DBC as the source database in a **COPY FROM** statement, but specify a target database name that is different than DBC. This allows you to copy DBC from a source system to a different database on a target system. When used as the source database in a **COPY FROM** statement, database DBC is not linked to database SYSUDTLIB. Therefore, only database DBC is copied.

Do not use database SYSUDTLIB as the target database name in a **COPY** statement. You can use SYSUDTLIB as the source database in a **COPY FROM** statement, but specify a target database name that is different than SYSUDTLIB. This allows you to copy SYSUDTLIB from a source system to a different database on a target system. When used as the source database in a **COPY FROM** statement, database SYSUDTLIB is not linked to database DBC. Therefore, only database SYSUDTLIB is copied.

Teradata ARC supports duplicate rows in a copy operation. However, if a restart occurs during the copy of a table with duplicate rows, the duplicate rows involved in the restart may be removed.

When restoring a cluster archive to a reconfigured system, run the cluster archives serially in sequence. If this restore scenario is detected by ARCMAN, then the NO BUILD option is automatically enabled, if not specified, and an explicit **BUILD** command must be run after all restore jobs are completed.

The **COPY** statement moves an archive copy of a database or table to a new environment. Copy options let you rename archive databases or create tables in existing databases. The table does not have to exist in the database it is being copied into. However, because the Teradata

Database internal table identifier is different for the new table, access rights for the table are not copied with the data.

Do not copy a table name generated in a character set different from your default character set or the character set specified in your Teradata ARC invocation. For example, a table name from a multibyte character set archive cannot be copied into a single-byte character set database.

To work around this restriction:

- 1 ANALYZE the archive tape with the HEX runtime option specified.
- 2 Result: The table names are displayed in hexadecimal format (e.g., X' C2C9C72A750A3AC5E4C5E4C36DD9D6E6)
- 3 Run the **COPY** table statements using the hexadecimal table names.

To copy more than one table or database with one **COPY** statement, separate the names with commas. Because Teradata ARC reads the archive data for each **COPY** statement it processes, one **COPY** statement yields better performance than multiple **COPY** statements.

Example

In the next example, the first table is copied to a different table name, and the next two tables are copied with the same table name.

```
COPY DATA TABLES (test.test1) (FROM (oldtest.test1))
, (test2.test)
, (test3.test)
, RELEASE LOCK
, FILE = ARCHIVE;
```

If the configuration of the source and target platforms are different, you can copy cluster archives only to all AMPs or specific AMPs.

Use these options with all types of copy operations:

- FROM (*dbname.tablename*)
- RELEASE LOCK
- NO BUILD
- CLUSTER = *nnn*
- AMP = *n*
- FILE = *name*

These options are suitable only for data table copy operations, and only with dictionary and data table copies:

- WITH JOURNAL TABLE (*dbname.tablename*)
- NO FALLBACK
- NO JOURNAL

Journal copy options are not supported after the target system is reconfigured. Journal table copy options define target (receiving) tables for change image journaling.

Teradata ARC uses the journal table copy options to:

- Select the set of journal images from an archive to copy.
- Name the tables in the receiving system to which the images apply during rollforward or rollback operations.

When you specify NO BUILD, Teradata ARC leaves the restored journal in a prebuild state. Teradata ARC does not allow roll operations on journal tables in this state, and generates an error message if you attempt to roll this journal back or forward.

If you copy data tables into a database that does not already have tables with those names, Teradata ARC creates the required tables. This is true *only* for data tables. Journal tables can *never* be created by a copy operation. They must exist in the receiving database. Similarly, you cannot create receiving databases with a COPY operation.

The COPY function preserves table names without converting them to upper case and returns these table names in their original format.

Database DBC.DBCAssociation

Teradata ARC creates a row in database DBC.DBCAssociation table for each dictionary row Teradata ARC copies successfully. This row contains columns with information that associate the copied table back to its originating Teradata Database.

The row in database DBC.DBCAssociation also contains columns that link it to the TVM table, which associates the row to its receiving table, and to the DBase table, which associates the row to its receiving database.

Teradata ARC also creates a column in each row of database DBC.DBCAssociation that links them to the RCEvent table. This link is the restore event number. For successful copy operations, the EventType in the RCEvent table is COPY.

Locks

Copy places exclusive client utility locks during its operations. If you copy an entire database from a full database archive, the entire database is locked for the duration of that operation. If you copy a table level archive or a single table, Teradata ARC locks *only* that table.

Views, Macros, Stored Procedures, Triggers, UDFs, and UDTs

Copying a full database archive is the same as a restore operation. Teradata ARC drops all existing tables, views, macros, stored procedures, triggers, UDFs, UDTs, and dictionary information in the receiving system. The data in the archive is then copied to the receiving database.

However, triggers cannot be copied with the **COPY** statement. If a trigger is defined in a database, then <trigger> NOT COPIED is displayed when the **COPY** statement is executed. This is not an error or warning. Triggers must be manually recreated via SQL.

You cannot copy one or more stored procedures from one database to another using the **COPY** statement. They can only be copied as part of a full database.

If your views, stored procedures, and macros have embedded references to databases and objects that are not in the receiving environment, those views, stored procedures, and macros

will not work. To make any such views, stored procedures, and macros work, recreate or copy the references to which they refer into the receiving database.

If your views, stored procedures, and macros have embedded references to databases and objects that *are* in the receiving environment, they will work correctly.

Note: Fully qualify all table, stored procedure, and view names in a macro and all table names in a view. If you do not, you may receive an error. When you execute a **COPY** statement, partial names are fully qualified to the default database name. In some cases, this may be the name of the old database.

Referential Integrity

After an all-AMPs copy, copied tables do not have referential constraints. First, referential constraints are not copied into the dictionary definition tables, database DBC.ReferencedTbls and database DBC.ReferencingTbls, for either a referenced (parent) or referencing (child) table copied into a Teradata Database. Moreover, all referential index descriptors are deleted from the archived table header before it is inserted into the copied table.

Reference constraints remain, however, on any table for which the copied table is a referenced table (a parent table) or any table the copied table (as a child table) references. Hence, when either a referenced or a referencing table is copied, the reference may be marked in the dictionary definition tables as inconsistent.

While a table is marked as inconsistent with respect to referential integrity, no updates, inserts or deletes are permitted. The table is fully usable only when the inconsistencies are resolved.

- Use an **ALTER TABLE DROP INCONSISTENT REFERENCES** statement to drop inconsistent references.
- Use the **ALTER TABLE ADD CONSTRAINT FOREIGN KEY REFERENCES** to add referential constraints to the copied table.

For an all-AMPs copy, the **REVALIDATE REFERENCES FOR** statement is not applicable with respect to the **COPY** statement.

Using Keywords with COPY

The **COPY** statement is affected by the following keywords. For information about the keywords that apply to copying partitioned data, see [“Restores of Selected Partitions” on page 195](#).

NO FALLBACK Keywords

This option applies only during a copy of a dictionary archive or an all-AMPs archive.

If a fallback table has permanent journaling on the archive, the table has dual journaling of its non-fallback rows after the copy when Teradata ARC applies the NO FALLBACK option (unless NO JOURNAL is specified).

FROM Keyword

The object specified in the FROM keyword identifies the archive object. This option applies only during a copy of a dictionary archive or an all-AMPs archive.

Journal enabled tables in the original archive carry their journaling forward to the copy unless you specify the NO JOURNAL keywords.

The NO JOURNAL keywords apply to all tables in a database when you copy a database. This option has no effect on a target database's journaling.

If the object you specify in the FROM option is a table, Teradata ARC copies only that table.

If the object you specify in the FROM option is a database, one of the following occurs:

- If the receiving object is a table, then Teradata ARC searches through the archived database for the named table and copies the table.
- If the receiving object is a database, then Teradata ARC copies the entire database into the receiving database.

Although you cannot copy into database DBC, you can specify database DBC as the object of the FROM option. In other words, you can copy database DBC into another database.

Similarly, you cannot copy into database SYSUDTLIB, but you can specify database SYSUDTLIB as the object of the FROM option. That is, you can copy database SYSUDTLIB into another database.

If you specify both the FROM and EXCLUDE TABLES options in the same statement, specify the FROM clause first. Then ARC can generate a default source database name for the excluded tables if a database name is not specified in the EXCLUDE TABLES clause.

WITH JOURNAL TABLE Keywords

This option only applies during a copy of a dictionary archive or an all-AMPs archive. To use this option, you must have INSERT access rights to the referenced journal table. The source table must have a journal or this option has no effect.

If you are copying a database, the journaling you specify with this option applies only to those tables that had journaling enabled in the original database. This option has no effect on a receiving database's default journaling.

If the database has default journaling specified, then Teradata ARC uses those options. This option only overrides the journal table in the *receiving* database, and is only valid if the originating table had journaling enabled.

If you want to carry journaling from the original table forward, but you do not specify the WITH JOURNAL TABLE option, Teradata ARC uses one of the following:

- The default journal of the receiving database if the table is to be created. If there is no default journal, Teradata ARC rejects the copy.
- The journal you specified in the WITH JOURNAL option when you created the table.
- The journal of the table being replaced if the table already exists.

APPLY TO Keywords

This option is required when copying journal images. Restore access rights to each table you want to specify are necessary. You can apply journal images to as many as 4096 tables, however, all of the table names in the APPLY TO option cannot exceed 8 KB.

When you copy a journal table, Teradata ARC restores only checkpoint rows and those rows that apply to the receiving environment. The tables to which those images apply and their journaling options must have already been copied. Thus, to perform a **COPY JOURNAL** with an **APPLY TO** option, the appropriate **COPY DATA TABLE** statement must have been issued at least once prior to issuing a **COPY JOURNAL** statement to establish the source/target tableid mapping in database DBC.DBCAssociation table. Teradata ARC copies only those images that have the correct table structure version (as defined by the originating table).

If you apply a journal image for the source table to more than one target table, Teradata ARC writes two occurrences of the journal image into the restored journal.

CLUSTER Keyword

This option is valid only if the source archive is a cluster archive and the copy is to the *identical* Teradata Database. The option is useful in restoring or copying a dropped table to a different database on the same Teradata Database.

AMP Keyword

With Teradata Database, use the **AMP=*n*** option to specify up to five AMPs.

Only the **NO FALLBACK TABLE** or **JOURNAL TABLE** options are applicable. It is useful following all-AMPs copies where all AMPs were not available at the time of the copy operation and non-fallback or journal receiving tables were involved.

NO BUILD Keywords

Use the **NO BUILD** keywords for all archives except the last one. Omit the **NO BUILD** option for the last archive, so that Teradata ARC can:

- Build indexes for tables.
- Sort change images for journal tables.

If the **NO BUILD** keywords are used during a **RESTORE** or **COPY** statement, run a separate **BUILD** statement for all databases and/or tables that were restored. The tables are not accessible until a **BUILD** statement is run.

ALL FROM ARCHIVE Keywords

The **ALL FROM ARCHIVE** keywords take the place of the database and/or table names that are normally specified after the **DATA**, **DICTIONARY**, **JOURNAL**, or **NO FALLBACK TABLES** keywords.

Do not specify any other database or table names to be copied when using **ALL FROM ARCHIVE**. Exercise caution when using this option, as all databases and tables in the given archive file will be copied, and any existing databases or tables will be overwritten.

CATALOG and **Fastpath** are not supported while using **ALL FROM ARCHIVE**. If **CATALOG** (or **Fastpath**) is enabled when **ALL FROM ARCHIVE** is specified, it will be disabled and a warning message will be given.

ALL FROM ARCHIVE cannot be used to copy database DBC, and database DBC must be excluded by the user if it is present in the archive being copied.

The **COPY** statement supports the **EXCLUDE** option; the syntax and function of the option is identical to the **RESTORE** version of **EXCLUDE**. Unlike **RESTORE**, **COPY** only supports the **EXCLUDE** option when using **ALL FROM ARCHIVE**.

The **FROM**, **APPLY TO**, and **WITH JOURNAL TABLE** object options are not allowed when using **ALL FROM ARCHIVE**.

EXCLUDE TABLES Keyword

Use the **EXCLUDE TABLES** option to specify one or more tables to skip during a database-level **COPY**. Teradata ARC does not copy any dictionary or data for the specified tables. If any of the tables already exist in the target database, they are unaffected by the **COPY** operation.

Also specify the **EXCLUDE TABLES** option to retain an existing table during a database-level **COPY** if that table was not archived (or if it was excluded during an **ARCHIVE**). If you fail to specify the **EXCLUDE TABLES** option, the table will be dropped and replaced with the data (if any) stored in the archive.

If you specify both the **FROM** and **EXCLUDE TABLES** options in the same statement, specify the **FROM** clause first. Then ARC can generate a default source database name for the excluded tables if a database name is not specified in the **EXCLUDE TABLES** clause.

Copying Partitioned Data

You can copy selected partitions of PPI tables, meaning that you can back up of one or more partitions of a table so you can archive, restore, and copy only a subset of data in a table.

Before attempting to copy selected partitions, read [“Potential Data Risks When Archiving/Restoring Selected Partitions” on page 33](#).

Restrictions on Copying Partitioned Data

Normally, the table you want to copy does not need to exist in the target database. However, if you are going to copy selected partitions to the table, the table does need to exist in the target database. Additionally, the target table must be a full table and not just consist of selected partitions.

Copying selected partitions is not allowed to a table that has undergone any of the following major DDL changes:

- Changing primary index columns
- Changing from PPI to NPPI, or visa versa
- Adding or changing referential integrity constraints

Other restrictions exist for archiving selected partitions. See [“Restrictions on Archiving Selected Partitions” on page 146](#).

Keywords for Copying Selected Partitions

The options available for copying selected partitions include:

- **PARTITIONS WHERE**
- **LOG WHERE**

- ERRORDB/ERRORTABLES
- ALL PARTITIONS
- QUALIFIED PARTITIONS

PARTITIONS WHERE Keyword

Use the PARTITIONS WHERE option to specify the conditional expression for selected partitions. The expression must contain the definition of the partitions that you want to copy. The following restrictions apply to the use of PARTITIONS WHERE:

- The object is an individual table (not a database).
- The source and target tables have a PARTITIONS BY expression defined.
- The copy is an all-AMP copy (not a dictionary, cluster, or journal copy).
- If the table belongs to a database that is specified in the **COPY** statement, the table is excluded from the database-level object (with EXCLUDE TABLES) and is individually specified.
- Any name specified in the conditional expression is within the table being specified. (Using table aliases and references to databases, tables, or views that are not specified within the target table result in an error.) It is recommended that the only referenced columns in the conditional expression be the partitioning columns or system-derived column PARTITION of the table. References to other columns does not contribute to partition elimination, and might accidentally qualify more partitions than intended.

LOG WHERE Keyword

If the PARTITIONS WHERE option does not capture all the rows that need to be copied, use the LOG WHERE option. This option inserts into a Teradata-generated error table archived rows that both fall outside the partitions specified by the PARTITIONS WHERE conditional expression *and* match the LOG WHERE conditional expression.

Use the option only if PARTITIONS WHERE is also specified for the object. If LOG WHERE is omitted, the default is to log to the error table only the rows in the partitions being restored that have errors.

ERRORDB/ERRORTABLES Keyword

The ERRORDB and ERRORTABLES options are mutually exclusive; specify only one option for an object. Also, specify either the PARTITIONS WHERE or ALL PARTITIONS option when using either ERRORDB or ERRORTABLES.

- If you specify ERRORTABLES without a database name, or if neither ERRORTABLES nor ERRORDB is specified, the error table is created in the same database as the base table.
- If you do not specify ERRORTABLES, by default the naming convention for the error table is the name of the base table plus the prefix “RS_”. For example, the error table for a table named “DataTable” is “RS_DataTable.” Names are truncated to 30 bytes.

ALL PARTITIONS Keyword

Use the ALL PARTITIONS option to copy all of the archived partitions in a table. These restrictions apply:

- The object being copied is an individual table, or the ALL FROM ARCHIVE option is specified.
- The source and target tables have a PARTITIONS BY expression defined.
- The copy is an all-AMP copy rather than a dictionary, cluster, or journal copy.
- PARTITIONS WHERE is not specified for the object.
- The partition bounding condition must have been well-defined when the backup was performed. A bounding condition is well-defined if the PARTITION BY expression on the source table consists of a single RANGE_N function, and if the specified range does not include NO RANGE or UNKNOWN. (Use ANALYZE to determine whether a selected partition is well-defined.)

If a conditional expression is not well-defined, Teradata ARC issues an error, and you must use PARTITIONS WHERE for the restore operation rather than ALL PARTITIONS.

QUALIFIED PARTITIONS Keyword

Use this option only to copy a specific-AMP archive after copying selected partitions from an all-AMP archive that was done while an AMP is down.

Alternate Character Set Support

Teradata ARC runs on IBM mainframes, which use the kanjiEBCDIC character set. However, object names created in kanji character sets (UNIX or MS/DOS platforms) other than EBCDIC can be copied from archive, but not translated into internal RDBMS format names.

The reason for this is that “hybrid” EBCDIC and kanjiEUC, or EBCDIC and kanjiShiftJIS single byte and multibyte characters cannot be translated into internal RDBMS format. If the object name is in internal RDBMS format, the Teradata Database can translate it into an external client format name.

Note: Hybrid object names result when archiving an object created in the KANJIEUC or KANJISJIS character set from a KANJIEBCDIC mainframe client and the hexadecimal representation of the object being archived is not used. In such a circumstance, the Teradata Database cannot properly translate the character set and produces a hybrid object name composed of single byte characters from the EBCDIC character set and multibyte characters from the creating character set (UNIX or DOS/V). The hybrid object name cannot be understood by the Teradata Database. Correct this in one of two ways:

- If you know the internal hexadecimal object name, in format (?.....?XN), use it. This is the *preferred* method.

The ARCMAN program can use Teradata Database to translate the internal hexadecimal format names to external hexadecimal format for the purpose of searching the archive file.

- If you do not know the internal hexadecimal object name:
 - a** Run the **ANALYZE** statement on the archive tape *without* making a connection to the Teradata Database by means of the **LOGON** statement.
 - b** The **ANALYZE** statement identifies the database in a “hybrid” format of the name.
 - c** Use the “hybrid” format of the name in the “FROM” syntax to specify that the object name in the archive file is different from the name of the target object name.

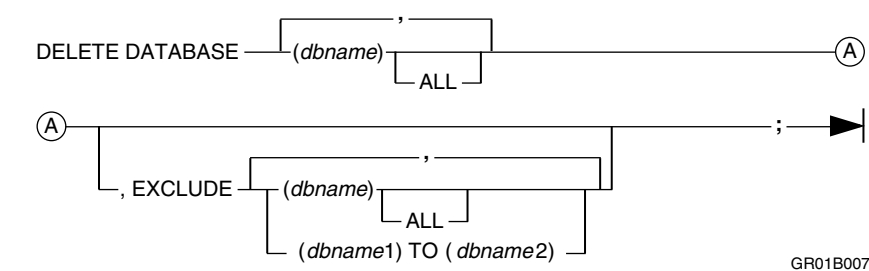
DELETE DATABASE

Purpose

The **DELETE DATABASE** statement deletes the contents of one or more databases, including all data tables, views, macros, stored procedures, and triggers.

To drop a journal table contents and structure, use the SQL statement **MODIFY DATABASE** with the DROP JOURNAL TABLE option. Refer to *SQL Reference: Statement and Transaction Processing* for more information.

Syntax



where

Syntax Element	Description
<i>(dbname)</i>	Name of the database.
ALL	Indicates the operation affects the named database and its descendants.
EXCLUDE	Protects the named database from deletion.
<i>(dbname)</i>	Name of the database to protect from deletion.
ALL	Indicates the operation protects the named database and its descendants.
<i>(dbname1) TO (dbname2)</i>	Alphabetical list of databases to protect from deletion. The delimiting database names need not identify actual databases. Database DBC is not part of a range.

Usage Notes

To delete a database, the user name specified in the **LOGON** statement must have DROP privilege on the database to be deleted.

The **DELETE DATABASE** statement is useful if you need to delete a database that is partially restored but unusable. Executing the statement deletes all access rights for tables, views, macros, stored procedures, triggers, and UDFs in the database. To release the locks that remain after the restore, execute the **RELEASE LOCK** statement before you enter the **DELETE DATABASE** statement.

Deleting All Objects from the Teradata Database

In order to restore database DBC, no objects can exist in any database outside of DBC. Delete all objects outside of DBC prior to restoring DBC by using this command:

```
DELETE DATABASE (DBC) ALL, EXCLUDE (DBC);
```

To drop permanent journal tables, use either the **MODIFY USER** or **MODIFY DATABASE** statement (See the MODIFY DATABASE/USER description in *SQL Reference: Data Definition Statements*.)

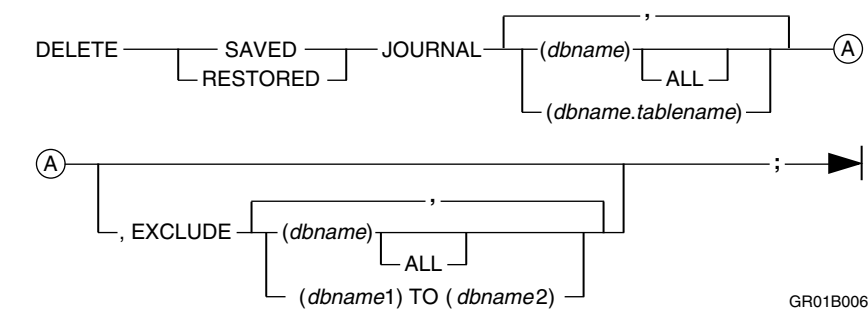
Enter the command exactly as shown above. You do not need to specify SYSUDTLIB as an object of this **DELETE DATABASE** statement because ARC removes the usual link between SYSUDTLIB and DBC so that only DBC is excluded from the delete operation. SYSUDTLIB is deleted so that DBC can be restored, however, it is deleted after all other databases have been deleted. This allows any objects that have definitions based on UDTs stored in SYSUDTLIB to be deleted before the UDTs themselves are deleted.

DELETE JOURNAL

Purpose

The **DELETE JOURNAL** statement removes a journal subtable from the Teradata Database.

Syntax



where

Syntax Element	Description
SAVED	Deletes the portion of the current journal that has previously been saved with a CHECKPOINT JOURNAL statement.
RESTORED	Deletes the portion of a journal table that was restored from archive for use in a recovery activity.
(dbname)	Name of the database that contains the journal to delete.
ALL	Indicates the operation affects the named database and its descendants.
(dbname.tablename)	Journal table within the named database to delete. All tables should be distinct within the list. Duplicate tables are ignored.
EXCLUDE	Protects the named journal tables from deletion.
(dbname)	Name of the protected journal table.
ALL	Keyword to protect the named journal and all its descendants.
(dbname1) TO (dbname2)	Alphabetical list of databases to be protected from deletion. The delimiting database names need not identify actual databases. Database DBC is not part of a range.

Access Privileges

To delete a journal table, the user name specified in the **LOGON** statement must have one of the following:

- The RESTORE privilege on the database or journal table being deleted
- Ownership of the database that contains the journal table

Usage Notes

You cannot use the journal archive to recover all AMPs when all of the following conditions exist:

- The checkpoint with an access lock creates a saved journal.
- The journal is not a dual journal.
- AMPs are offline.

Because transactions between the all-AMPs archive and the single-AMP archive may not be consistent, you cannot delete a saved journal with an AMP offline that has no dual journal.

ENABLE DATA EXTENSION

Purpose

The **ENABLE DATA EXTENSION** statement loads an extension module that allows archive data to be processed (for example, encrypted). ARC processes the data before sending it to the output media.

Syntax

```
ENABLE DATA EXTENSION — , — MODULE — = — ' — modulename — ' — , — PARM — = — ' — parameters — ' — ; —▶|
```

2412A027

where

Syntax Element	Description
<i>modulename</i>	File name of the extension module to load. The file extension for the file name is optional. If the extension is not specified, the correct shared library extension for the platform is added (for example, <i>.dll</i> for Windows).
<i>parameters</i>	Parameters that are required for the module.

Usage Notes

Note: This statement is valid only on Windows platforms.

Data extensions affect the processing of one or more ARC statements and must be specified before the ARC statement so that the data extension will be active by the time processing of the ARC statement begins. The data extension determines which ARC statements will be affected.

The *modulename* and PARM elements are both required when specifying **ENABLE DATA EXTENSION** because they specify which data extension is used, and the parameters that are applied to that data extension. After a data extension is enabled, it remains active until the end of the archive operation.

Currently, the encryption data extension is supported. The data encryption module is used during an archive operation to encrypt the data read from the Teradata Database before the data is stored in the archive file. The encryption data extension is currently provided by

Protegrity. The Protegrity data encryption module, *pepbar.plm*, supports the AES128, AES256, and PANAMA data encryption algorithms.

When data in an encrypted archive file is restored, copied, or analyzed, the data encryption module and algorithm that was used to originally encrypt the data is automatically loaded by ARC to decrypt the data after it is read from the archive file. The data can then be processed as unencrypted data.

To enable the Protegrity data encryption module using the AES 256-bit algorithm, specify:

```
ENABLE DATA EXTENSION, MODULE='pepbar.plm', PARM='ALGORITHM=AES256';
```

It is also possible to specify the Protegrity data encryption module by using **ENABLE ENCRYPTION**, a shorthand method for specifying a data encryption extension. For more information about data encryption extensions, see [“ENABLE ENCRYPTION” on page 174](#).

ENABLE ENCRYPTION

Purpose

The **ENABLE ENCRYPTION** statement loads a module that allows archive data to be encrypted before sending it to the output media.

Syntax

```
ENABLE ENCRYPTION [ , -MODULE = ' -modulename - ' ] [ , -ALGORITHM = ' -algoname - ' ] ;
```

2412A028

where

Syntax Element	Description
<i>modulename</i>	[Optional] File name of the encryption extension module to load. If the name is not specified, the Protegrity encryption module, <i>pepbar.plm</i> , is used. The file extension for the file name is optional. If the extension is not specified, the correct shared library extension for the platform is added (for example, <i>.dll</i> for Windows).
<i>algoname</i>	[Optional] The algorithm supported by the encryption module. If the algorithm is not specified, a default of AES128 is used.

Usage Notes

Note: This statement is valid only on Windows platforms.

Data extensions affect the processing of one or more ARC statements and must be specified before the ARC statement so that the data extension will be active by the time processing of the ARC statement begins. The data extension determines which ARC statements will be affected.

ENABLE ENCRYPTION is a shorthand method for specifying a data encryption extension. After a data encryption extension is enabled, it remains active until the end of the archive operation.

Data encryption modules are used during archive operations to encrypt the data read from the Teradata Database before the data is stored in the archive file.

The encryption data extension is currently provided by Protegrity. The Protegrity data encryption module, *'pepbar.plm'*, supports the AES128, AES256, and PANAMA data encryption algorithms.

When data in an encrypted archive file is restored, copied, or analyzed, the data encryption module and algorithm that was used to originally encrypt the data is automatically loaded by ARC to decrypt the data after it is read from the archive file. The data can then be processed as unencrypted data.

Example

```
ENABLE ENCRYPTION;
```

In this code example, the *pepbar.plm* Protegrity encryption module and AES128 encryption algorithm are used because they are the defaults.

Example

```
ENABLE ENCRYPTION, MODULE='pepbar.plm';
```

This code example specifies the Protegrity encryption module, *pepbar.plm*, and uses the default AES128 encryption algorithm.

Example

```
ENABLE ENCRYPTION, ALGORITHM='AES256';
```

This code example specifies the AES256 encryption algorithm, and uses the default Protegrity encryption module, *pepbar.plm*.

Example

```
ENABLE ENCRYPTION, MODULE='pepbar.plm', ALGORITHM='PANAMA';
```

This code example specifies the PANAMA encryption algorithm and the Protegrity encryption module, *pepbar.plm*.

Note: Use the complete name, *pepbar.plm*, when specifying the Protegrity data encryption module. Specifying `MODULE='pepbar'` without the file extension generates an invalid module name for the Protegrity data encryption module because the default file extension is applied to *pepbar* (for example, *dll*).

For documentation relating to the Protegrity module, go to <http://www.protegrity.com/solutioncenter.html>. The *Defiance BAR Encryption Administration Guide*:

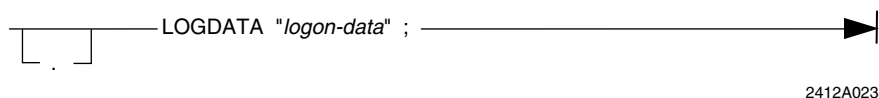
- Summarizes the Defiance BAR Encryption components
- Describes installation and configuration of Defiance BAR Encryption on Teradata BAR servers in a Windows environment
- Describes the Defiance BAR Encryption activation steps

LOGDATA

Purpose

This statement specifies any additional logon or authentication data that is required by an extended security mechanism.

Syntax



Usage Notes

The format of data in the **LOGDATA** statement varies, depending on the logon mechanism being used. However, to take effect, the **LOGDATA** statement must be specified (with a **LOGMECH** statement) *before* the **LOGON** statement.

The data that is required by the logon mechanism is defined as *logon-data*, as seen in the syntax diagram above. This *logon-data* must be valid for all of the Teradata ARC sessions that will be logging on. For most logon mechanisms, *logon-data* is case-sensitive, and it must be enclosed in double quotes if it contains any whitespace characters or non-alphanumeric characters.

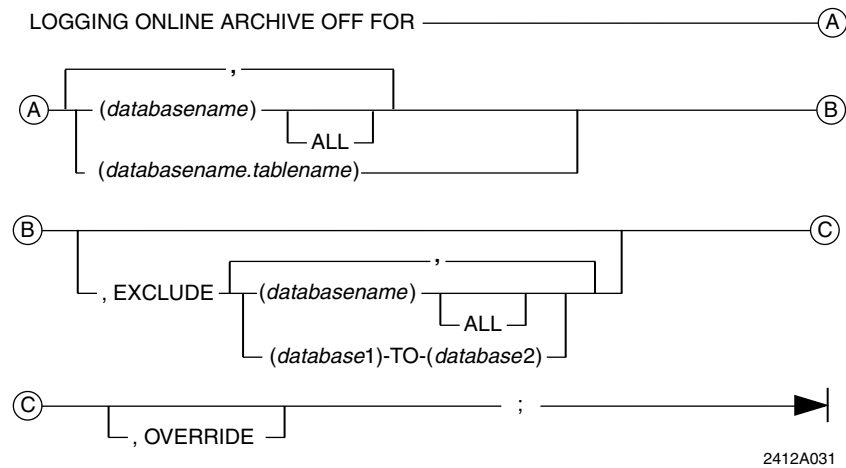
For more information about extended security mechanisms and the **LOGDATA** statement, see *Teradata Call-Level Interface Version 2 Reference for Network-Attached Systems*.

LOGGING ONLINE ARCHIVE OFF

Purpose

LOGGING ONLINE ARCHIVE OFF ends the online archive process.

Syntax



where

Syntax Element	Description
<i>database</i>	Indicates the database that is included in online archive logging.
ALL	[Optional] Indicates that the specified database and all of its child databases are included in the statement.
<i>databaseName.tableName</i>	Indicates the table and database that is included in online archive logging.
EXCLUDE	[Optional] Specifies the databases that are excluded from the statement.
OVERRIDE	[Optional] Allows online archive logging to be stopped by someone other than the user who initiates the process.

Usage Notes

An online archive is a process in which rows are archived from a table at the same time update, insert, or delete operations on the table are taking place. When an online archive is initiated on a table or a database, Teradata ARC creates a log for the specified table or a separate log for each table in the specified database. **LOGGING ONLINE ARCHIVE OFF** ends the online

archive. The log, which contains all changes to the table, is archived as a part of the archive process. When the table is restored or copied, the changes recorded in the table's log are used to roll back the changes in the table. In other words, the table is rolled back to the point that was established when online archive logging was started on that table.

LOGGING ONLINE ARCHIVE OFF is used after the archive process has completed. The statement explicitly ends online archive logging on the tables and databases that were just archived. When the online archive logging process is terminated, the logging of changed rows is stopped, and the log is deleted.

If the online archive logging process is not terminated on a table after the table has been archived, logging continues, consuming system resources such as CPU cycles and perm space. In time, it is possible to run out of perm space, causing the Teradata Database to crash.

Example

This example stops online archive logging on Table1.

```
LOGGING ONLINE ARCHIVE OFF FOR (DatabaseA.Table1);
```

Example

This example stops online archive logging on all the tables in the DatabaseA.

```
LOGGING ONLINE ARCHIVE OFF FOR (DatabaseA);
```

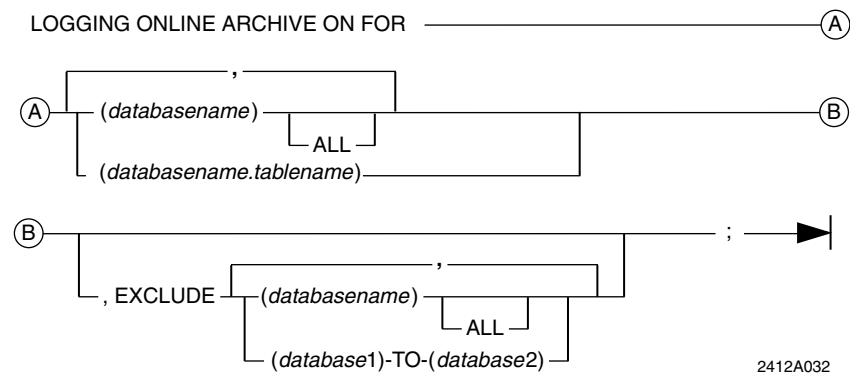
LOGGING ONLINE ARCHIVE ON

Purpose

LOGGING ONLINE ARCHIVE ON begins the online archive process.

Use this statement for cluster archives.

Syntax



where

Syntax Element	Description
<i>database</i>	Indicates the database that is included in online archive logging.
ALL	[Optional] Indicates that the specified database and all of its child databases are included in the statement.
<i>database.table</i>	Indicates the table and database that is included in online archive logging.
EXCLUDE	[Optional] Specifies the databases that are excluded from the statement.

Access Privileges

To execute **LOGGING ONLINE ARCHIVE ON**, the user name specified in the logon statement must have one of these privileges:

- Archive privilege on the database or table that being logged. The privilege is granted to a user or an active role for the user.
- Ownership of the database or table.

Usage Notes

An online archive is a process in which rows are archived from a table at the same time update, insert, or delete operations on the table are taking place. When an online archive is initiated on a table or a database, Teradata ARC creates a log for the specified table or a separate log for each table in the specified database. The log, which contains all changes to the table, is saved as a part of the archive process. The changes recorded in the log can be applied to any changes to the table that occurred during the archive, such as a restore operation.

EXAMPLE

This example starts online archive logging on Table1.

```
LOGGING ONLINE ARCHIVE ON FOR (DatabaseA.Table1);
```

EXAMPLE

This example starts online archive logging on Table1, Table2 and Table3.

```
LOGGING ONLINE ARCHIVE ON FOR  
(DatabaseA.Table1),  
(DatabaseA.Table2),  
(DatabaseA.Table3);
```

Use **LOGGING ONLINE ARCHIVE ON** to start online archive logging for specified objects before submitting an archive job for those objects. After logging is started for a target table, any changed rows to that table are recorded into a log that grows until the database runs out of space or **LOGGING ONLINE ARCHIVE OFF** is used to end online archive logging.

The total number of tables that can be in an active state of online archive logging at the same time is 10000.

When the table is restored or copied, the changes recorded in the log are used to roll back those changes in the table to the point that was established when online archive logging was started on that table.

Disallowed Tables

There are restrictions on the types of tables used with **LOGGING ONLINE ARCHIVE ON**. Do not specify these tables, or the online archive process will abort:

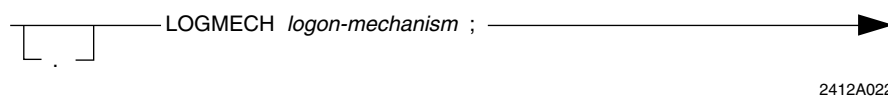
- Temporary tables
- Permanent journal tables
- Fastload aborted tables
- Multiload aborted tables
- Multiload work tables
- Online archive logging activated tables

LOGMECH

Purpose

The **LOGMECH** statement sets the extended security mechanism used by Teradata ARC to log onto a Teradata Database.

Syntax



Usage Notes

The **LOGMECH** statement must be specified with the **LOGDATA** statement *before* the **LOGON** statement in order to take effect.

As seen in the syntax diagram, above, *logon-mechanism* is the name of the logon mechanism that you want to use for a specific archive job. The specified logon mechanism is then used for all Teradata ARC sessions that will be logging on.

For a complete list of supported logon mechanisms, see *Teradata Call-Level Interface Version 2 Reference for Network-Attached Systems*.

LOGOFF

Purpose

The **LOGOFF** statement ends all Teradata Database sessions logged on by the task and terminates Teradata ARC.

Syntax



Usage Notes

When Teradata ARC accepts the **LOGOFF** statement, it issues this message and ends the session:

ARCMAN HAS LOGGED OFF n SESSIONS

Teradata ARC control language parser ignores any statements following the **LOGOFF** statement.

LOGON

Purpose

The **LOGON** statement specifies the name of the Teradata machine that Teradata ARC should connect to, as well as the user name and password that should be used.

Note: A LOGON string that does not contain a *userid* and a *password* is interpreted as a SSO (single sign-on) logon.

Syntax

```
LOGON [tdpid/] [userid] [, password] [, 'acctid'];
```

2412B006

where

Syntax Element	Description
<i>tdpid</i>	<p>The identifier of the Teradata machine that will be used for this job.</p> <p>Note: <i>tdpid</i> is followed by a “/”.</p> <p>The <i>tdpid</i> must be a valid identifier configured for your site.</p> <p>If you do not enter a value, <i>tdpid</i> defaults to the id established by the system administrator.</p>
<i>userid</i>	<p>User identifier.</p> <p>A <i>userid</i> can be up to 30 characters.</p> <p>Your <i>userid</i> must be authorized to perform the operations specified by subsequent utility statements.</p> <p>Note: This field is optional if single sign-on (SSO) is implemented</p>
<i>password</i>	<p>Password associated with the user name.</p> <p>A password can be up to 30 characters.</p> <p>If you use a null password, include a comma before the semicolon.</p> <p>Note: This field is optional if Single Sign-On (SSO) is implemented</p>
<i>'acctid'</i>	<p>Account identifier associated with the <i>userid</i>.</p> <p>If you omit this value, Teradata ARC uses the default account identifier defined when your <i>userid</i> was created.</p>

Usage Notes

The **LOGON** statement causes Teradata ARC to log on only two control sessions for the user.

When Teradata ARC encounters an **ARCHIVE**, **RESTORE**, or **COPY** statement, it automatically logs on any additional data sessions specified in the **SESSIONS** runtime parameter. **ARCHIVE**, **RESTORE**, and **COPY** statements are the only statements that need these sessions.

The **LOGON** statement establishes sessions by:

- Identifying the user to the Teradata Database
- Specifying the account to charge for system resources

When the **LOGON** statement is accepted, this message is displayed:

```
2 SESSIONS LOGGED ON
```

If the user who is specified by *userid* is logged onto the Teradata Database through a program other than Teradata ARC, Teradata ARC terminates as soon as it encounters a **BUILD**, **COPY**, **RESTORE**, **ROLLBACK**, or **ROLLFORWARD** statement. To determine whether a user is logged onto the Teradata Database, Teradata ARC performs a **SELECT** on database `DBC.sessionInfo`. Therefore, the user attempting to log on must have **SELECT** privileges on `DBC.sessionInfo`.

If a **LOGON** statement has already been executed during the run of Teradata ARC, Teradata ARC logs off all Teradata Database sessions from the previous logon.

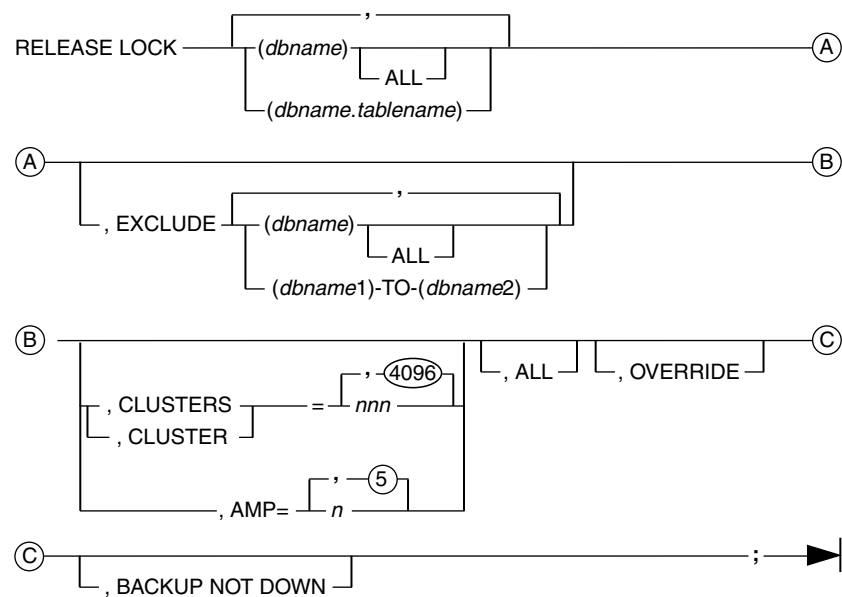
The implementation of the SSO feature provides the ability to log onto a workstation once and then access the Teradata Database without having to provide a user name and password. SSO saves the time required to enter these fields, which are now optional. Additionally, certain authentication mechanisms (e.g., Kerberos, NTLM) do not send passwords across the network to further heighten security. SSO requires support of both the database and the client software.

RELEASE LOCK

Purpose

The **RELEASE LOCK** statement removes a utility (HUT) lock from the identified databases or tables.

Syntax



2412B013

where

Syntax Element	Description
<i>(dbname)</i>	Name of the database on which the HUT lock is to be released.
ALL	Indicates the operation affects the named database and its descendants.
<i>(dbname.tablename)</i>	Name of the table on which the HUT lock is to be released.
EXCLUDE	Prevents HUT locks on the named databases from being released.
<i>(dbname)</i>	Name of database to exclude from release.
ALL	Excludes the named database and its descendants.

Syntax Element	Description
<i>(dbname1)</i> TO <i>(dbname2)</i>	Alphabetical list of databases to exclude from release. The delimiting database names need not identify actual databases. Database DBC is not included as part of a range.
CLUSTERS = <i>nnn</i> or CLUSTER = <i>nnn</i>	Specifies AMP cluster for which locks are released. <i>nnn</i> is the number of the cluster on which locks are released. Specify up to 4096 cluster numbers. Locks are not released on any offline AMPs in the specified clusters unless you specify the ALL option.
ALL	Releases locks on AMPs that are offline when the RELEASE LOCK statement is issued. Teradata ARC releases HUT locks when the AMPs return to online operation. Note: You cannot use the ALL keyword with the AMP option.
OVERRIDE	Allows HUT locks to be released by someone other than the user who set them.
BACKUP NOT DOWN	Allows HUT locks to remain on non-fallback tables (with single after image journaling) for those AMPs where the permanent journal backup AMP is down. Teradata ARC releases all other HUT locks requested.

Access Privileges

To release the HUT locks on a database or a table in a database, the user name specified in the **LOGON** statement must have one of the following:

- The ARCHIVE or RESTORE privilege on the database or table where HUT locks are to be released
- Ownership of the database or table where HUT locks are to be released

When you specify the OVERRIDE keyword, the user name specified in the **LOGON** statement must have one of the following:

- The DROP DATABASE privilege on the specified database or table
- Ownership of the database or table

Usage Notes

During an archive or recovery operation, Teradata ARC places HUT locks on the objects affected by the operation. These locks remain active during a Teradata Database or client restart, and must be explicitly released by the **RELEASE LOCK** statement or by the RELEASE LOCK keywords available on the **ARCHIVE**, **REVALIDATE REFERENCES FOR**, **ROLLBACK**, **ROLLFORWARD**, **RESTORE** and **BUILD** statements. Teradata ARC issues a message to report that the release operation is complete. It releases HUT locks when the AMPs return to online operation.

If you specify the user name in the logon statement, the **RELEASE LOCK** statement or the equivalent option in a utility statement releases all HUT locks placed on the specified database or table. If you specify the **OVERWRITE** keyword in the **RELEASE LOCK** statement, Teradata ARC releases all HUT locks on the specified database or table.

If HUT locks were placed at the database-level, they must be released at the database-level.

A table-level release lock against a database-level lock has no effect. Conversely, a database-level release lock releases all locks on the database, including any database-level lock and all table level locks.

You cannot specify **ALL** with the **AMP** keyword.

Warning: Releasing a HUT lock while another Teradata ARC job is running could result in data corruption and/or unexpected errors from ARCMAN or the Teradata Database.

BACKUP NOT DOWN Keywords

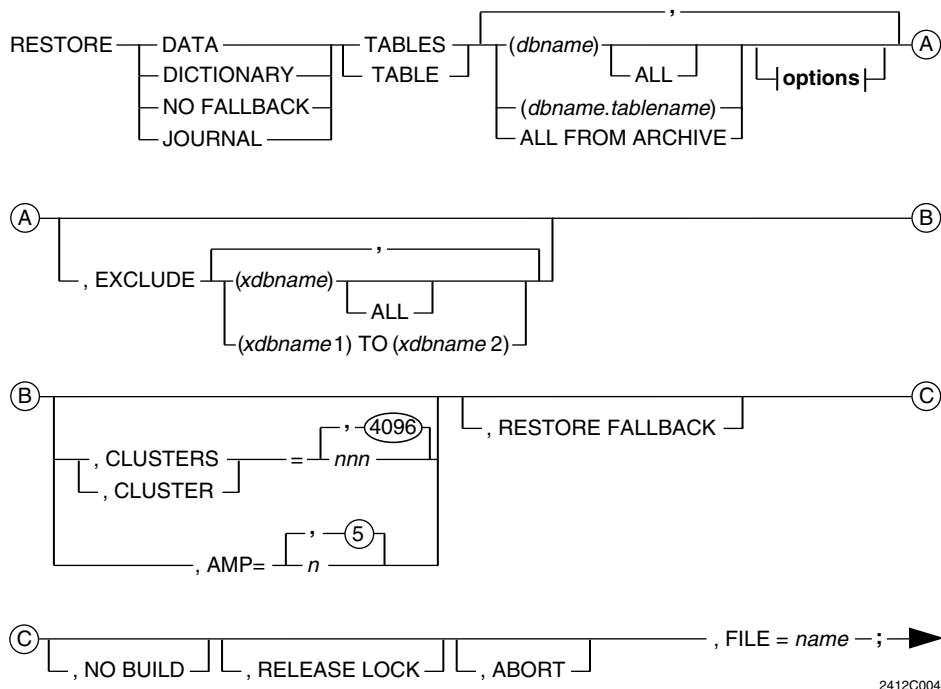
With Teradata Database, **BACKUP NOT DOWN** allows HUT locks to remain on non-fallback tables with remote single after image journaling when the backup AMP is offline.

RESTORE

Purpose

The **RESTORE** statement moves data from archived files to the same Teradata Database from which it was archived, or moves data to a Teradata Database other than the one used for the archive, if the database DBC is already restored. To use **RESTORE** to import data to the Teradata Database, the table being restored must exist on the target Teradata Database.

Syntax





2412A019

where

Syntax Element	Description
DATA TABLE or DATA TABLES	Restores both fallback and non-fallback tables to all AMPs or clusters of AMPs.
DICTIONARY TABLE or DICTIONARY TABLES	Restores only dictionary rows. This option is only allowed for: <ul style="list-style-type: none"> Archives created using the DICTIONARY option of the ARCHIVE statement All-AMPs journal archives
NO FALLBACK TABLE or NO FALLBACK TABLES	Restores tables without fallback to complete a previous all-AMPs or cluster restore where AMPs were offline, or to restore non-fallback data to specific AMPs recovering from disk failure.
JOURNAL TABLE or JOURNAL TABLES	Restores an archived journal for subsequent use in a roll operation.
ALL FROM ARCHIVE	Specifies that all databases and tables in the given archive file will be restored and that any existing databases and tables will be overwritten. This option is used in place of database/table names normally specified. No other database or table names can be specified to be restored with this option.
(dbname)	Name of the database to restore. This form restores all tables of the type specified, and all stored procedures (if specified type is FALLBACK) that are in the named database and on the input file.
ALL	Restores the named database and all descendants.

Syntax Element	Description
<i>(dbname.tablename)</i>	Name of a table within the named database to restore. All tables should be distinct within the list. Duplicate table names are ignored.
EXCLUDE	Prevents specified databases from being restored.
<i>(xdbname)</i>	Name of database to exclude from restore.
ALL	Excludes the named database and all descendants.
<i>(xdbname1)</i> TO <i>(xdbname2)</i>	Alphabetical list of client databases to exclude from the restore. The delimiting database names need not identify actual databases. Database DBC is not part of a range.
CLUSTERS = <i>nnn</i> or CLUSTER = <i>nnn</i>	Specifies AMP clusters to restore. <i>nnn</i> is the cluster number. Specify up to 4096 clusters.
AMP = <i>n</i>	Specifies the AMP (or a list of AMPs) to restore for Teradata Database, where <i>n</i> is the number of the AMP. Specify up to five AMPs with this option.
RESTORE FALLBACK	Restores the fallback copy of primary and unique secondary indexes while restoring the data. This option applies only to data table restores of fallback tables. Note: When the RESTORE FALLBACK option is specified and the system formats or hash functions of the source and target systems are different, then Teradata ARC will disable the RESTORE FALLBACK request.
NO BUILD	Prevents the Fallback and Secondary Index rows from being created. If NO BUILD is requested when restoring database DBC, the request will be ignored. If the NO BUILD keywords are used during a RESTORE statement, a separate BUILD statement must be run for all databases and/or tables that were restored. The tables will not be accessible until a BUILD statement is run.
RELEASE LOCK	Releases utility (HUT) locks on the named databases when the restore completes successfully.
ABORT	Aborts the restore if an AMP to which a non-fallback table is to be restored is not online. This option affects only an all-AMPs restore. It does not affect a specific AMP restore.
EXCLUDE TABLE or EXCLUDE TABLES	Prevents individual tables in the listed database from being restored.

Syntax Element	Description
<i>(tablename)</i>	Name of an individual table in the designated database to exclude. Multiple tables are separated by commas. This form (without a database name prefix) is used only when ALL has not been designated on the current object.
<i>(xdbname.tablename)</i>	List of fully qualified tables (i.e., prefixed by database name) to exclude.
PARTITIONS WHERE	Specifies the conditional expression for partition-level operations.
<i>(!conditional expression!)</i>	The conditional expression for specifying selected partitions.
LOG WHERE	Specifies the rows that are logged to an error table for manual insertion and deletion.
<i>(!conditional expression!)</i>	The conditional expression for specifying rows to log to the error table when restoring selected partitions
ALL PARTITIONS	Restores all archived partitions for an archived PPI object.
QUALIFIED PARTITIONS	Restores the same partitions specified in a previous selected-partition restore.
ERRORDB and ERRORTABLES	Specifies the location of the error log for partition-level operations.

Access Privileges

To restore a database or a data table within the database, the user name specified in the **LOGON** statement must have one of the following:

- The RESTORE privilege on the database or table that is being restored
- Ownership of the database or table

Only these users may restore database DBC:

- User DBC
- Users granted the RESTORE privilege on database DBC

To restore a journal table, the user name must have one of the following:

- The RESTORE privilege on the journal table itself or the database that contains the journal table
- Ownership of the database that contains the journal table

To use the restored journal in recovery activities also requires RESTORE privilege on the data tables being rolled forward.

Usage Notes

Database SYSUDTLIB is linked with database DBC and is only restored if DBC is restored. SYSUDTLIB cannot be specified as an individual object in a **RESTORE** statement.

If database DBC is involved in a restore operation, it is always restored first, followed by database SYSUDTLIB. If additional databases are being restored, they follow SYSUDTLIB in alphabetical order.

An all-AMPs restore or a specific AMP restore can use archive files created by any of the following:

- An all-AMPs archive
- A cluster archive
- A specific AMP archive
- An archive of selected partitions

During a restore of a Teradata Database, Teradata ARC issues messages in the following format as it restores tables, views, macros, stored procedures, triggers, UDFs and permanent journals:

```
"UDFname" - n,nnn,nnn BYTES, n,nnn,nnn ROWS RESTORED
"procedurename" - n,nnn,nnn BYTES, n,nnn,nnn ROWS RESTORED
"macroname" - MACRO RESTORED
"methodname" - METHOD RESTORED
"methodname" - n,nnn,nnn BYTES, n,nnn,nnn ROWS RESTORED
"tablename" - n,nnn,nnn BYTES, n,nnn,nnn ROWS RESTORED
"triggername" - TRIGGER RESTORED
"UDTname" - TYPE RESTORED
"viewname" - VIEW RESTORED
```

When all tables are restored, Teradata ARC issues this message:

```
STATEMENT COMPLETED
```

At the start of a restore, or after a system restart, all offline AMPs are listed in the output log.

After each restore, compare the contents of the output log with listings of tables that were restored to determine which tables were not restored to offline AMPs.

Teradata ARC supports duplicate rows in a restore operation. However, if a restart occurs during the restore of a table with duplicate rows, the duplicate rows involved in the restart might be removed.

Because unique secondary indexes are distributed across all AMPs, the **RESTORE** statement does *not* rebuild unique secondary indexes, even if the indexes are in a cluster that is not being restored. Use the **BUILD** statement to build and re-validate unique indexes.

All-AMP Restores

If you use an archive file from an all-AMPs archive for an all-AMPs restore of data tables, Teradata ARC deletes the current copy of the data tables being restored before it restores the archive copy.

Cluster AMP Restores

When restoring a cluster archive to a reconfigured system, run the cluster archives serially in sequence. If this restore scenario is detected by ARCMAN, the NO BUILD option is automatically enabled, if not specified, and an explicit **BUILD** command must be run after all restore jobs are completed.

If you use NO BUILD for a cluster restore operation, submit an all-AMPs **BUILD** statement after all the clusters are restored. Do not use the NO BUILD keywords for fallback table cluster restore operations. If a cluster loses an AMP before the all-AMPs build, you must restore that cluster manually.

If you use an archive file from a cluster archive for an all-AMPs restore of data tables, a dictionary restore must precede the all-AMPs data tables restore.

You cannot access fallback tables until you complete a build operation. You cannot update non-fallback tables until you complete a build operation.

Specific-AMP Restores

Use the specific-AMP restore to restore non-fallback data tables. Only perform this recovery when the AMP to be restored is online to normal Teradata Database operations. If you use an archive file from a specific-AMP archive, Teradata ARC does *not* delete the current copy of any non-fallback table being restored before it restores the archive copy.

If one or more AMPs are offline during archive or restore, it might be necessary to perform specific-AMP restores after you complete an all-AMPs or cluster restore. For example, if an AMP is online at the start of a restore and goes offline during the process, you might need to restore some non-fallback tables to the AMP when it comes back online. If, however, an AMP is offline at the start of a restore and comes back online before the procedure completes, Teradata ARC restores subsequent non-fallback tables to the AMP. In either case, specify the NO FALLBACK TABLES option of the **RESTORE** statement.

Journal Table Restores

Restore dictionary definitions of the journal tables with the **RESTORE DICTIONARY** statement against a journal archive. This creates empty journal tables.

If archive data sets containing journal tables are concatenated by some operation outside of the Teradata Database, subsequent journal tables are not accessed from the concatenation by the restore process.

Restore the definition of the journal table to a reconfigured system by performing a DICTIONARY RESTORE from an all-AMPs journal archive.

Table Restores

When you restore an entire table, restore an all-AMPs archive before any specific-AMP archive files. If you are restoring a single AMP from a disk failure, use the specific-AMP archive. Table restores accomplish the following:

- An all-AMPs restore restores all specified tables that also exist on the input archive file.
- An all-AMPs data table restore from an all-AMPs or dictionary tables archive file also restores the Data Dictionary entries for tables, views, macros, stored procedures, triggers, and UDFs with the databases.

Dictionary Table Restores

An all-AMPs dictionary tables restore must precede cluster restores involving fallback tables. A **DICTIONARY** restore leaves all fallback tables recovered in a state of restoring and leaves all indexes invalidated for non-fallback tables.

Restoring After a Disk Failure

To recover from a disk failure, run the Disk Copy and Table Rebuild utilities to restore database DBC and all user data tables defined with the fallback option, then perform a specific AMP restore.

Specify either database names or names of individual tables along with the **EXCLUDE** keyword to specify the non-fallback tables that were not restored while the AMP was inoperative.

When you cannot restore part of a non-fallback table during an all-AMPs restore because an AMP is offline, the system invalidates unique secondary indexes for the table. After the AMP is back online and the table has been restored, either drop and then recreate the unique secondary indexes for the table, or use the **BUILD** statement to recreate secondary indexes.

To restore from a catastrophic condition (for example, to restore database DBC) on a system that has permanent journaling, restore the system using an archive created from an **ARCHIVE JOURNAL TABLES** statement in the following order:

- 1 **RESTORE DATA TABLES** (DBC) ...;
 - 2 **RESTORE DICTIONARY TABLES** (DBC) **ALL**, **EXCLUDE** (DBC) ..., **FILE** = **ARCHIVE**;
- Note:** For this example, **ARCHIVE** is a journal archive.
- 3 **RESTORE DATA TABLES** (DBC) **ALL**, **EXCLUDE** (DBC) ...;
 - 4 **RESTORE JOURNAL TABLES** (DBC) **ALL**, **EXCLUDE** (DBC);

Note: Follow Step 4 only if you intend to roll forward or roll back the journal images.

The **RESTORE** statement builds the fallback copy of primary and non-unique secondary index subtables within a cluster. If the system fails during this process, **RESTORE** rebuilds fallback copies automatically.

Referential Integrity

When a table is restored into a Teradata Database, the dictionary definition of that table is also restored. The dictionary definitions of both the *referenced* (parent) and *referencing* (child) table contain the complete definition of a reference.

Nonetheless, in restoring tables it is possible to create an *inconsistent* reference definition in the Teradata Database. Hence, when either a referenced (parent) or a referencing (child) table is restored, the reference may be marked in the dictionary definition tables as inconsistent.

While a table is marked as inconsistent with respect to referential integrity, no updates, inserts or deletes are permitted. The table is fully usable only when the inconsistencies are resolved.

- If both the referenced (parent) and the referencing (child) tables are restored, use the **REVALIDATE REFERENCES FOR** statement to validate references. See [“REVALIDATE REFERENCES FOR” on page 202](#).

If inconsistent constraints remain after a **REVALIDATE REFERENCES FOR** statement has been executed, use the **ALTER TABLE DROP INCONSISTENT REFERENCES** statement to remove inconsistent constraints.

- If either the referenced (parent) table or the referencing (child) table is restored (but not both tables), the **REVALIDATE REFERENCES FOR** statement is not applicable. Use an **ALTER TABLE DROP INCONSISTENT REFERENCES** statement to drop the inconsistent references.

Restores of Selected Partitions

You can restore selected partitions of PPI tables. This allows you to archive and restore only a subset of data in a PPI table.

Restrictions on Restoring Selected Partitions

Before attempting to restore selected partitions, be sure to read [“Potential Data Risks When Archiving/Restoring Selected Partitions” on page 33](#) and [“Changes Allowed to a PPI Table” on page 147](#).

The following restrictions apply to restoring selected partitions:

- Restoring selected partitions is not allowed to a machine with a hash function that is different from the source machine, but a different configuration is allowed.
- To RESTORE or COPY selected partitions, a table must already exist on the target system. For COPY, the existing table must have been created by a full-table COPY from the source machine.
- Restoring selected partitions is not allowed to a table that has undergone any of the following major DDL changes:
 - Adding, modifying, or dropping columns.
 - Certain changes during RESTORE and COPY operations. For more information, see [“Restrictions on Copying Partitioned Data” on page 164](#).

Restoring selected partitions is not allowed to a table that has undergone any of the following major DDL changes:

- Changing primary index columns
- Changing from PPI to NPPI, or visa versa
- Adding or changing referential integrity constraints

Other restrictions exist for archiving selected partitions of PPI tables. For more information, see [“Restrictions on Archiving Selected Partitions” on page 146](#).

Keywords for Restoring Selected Partitions

These options are available for restoring selected partition archives:

- PARTITIONS WHERE
- LOG WHERE
- ERRORDB/ERRORTABLES
- ALL PARTITIONS
- QUALIFIED PARTITIONS

The next sections describe how to use the options for selecting partitions of PPI tables.

PARTITIONS WHERE Keyword

Use the PARTITIONS WHERE option to specify the conditional expression, which contains the definition of the partitions that you want to restore. The following restrictions apply to the use of PARTITIONS WHERE:

- The object is an individual table (not a database).
- The source and target tables have a PARTITIONS BY expression defined.
- The restore is an all-AMP restore (not a dictionary, cluster, or journal restore).
- If the table belongs to a database that is specified in the **RESTORE** statement, the table is excluded from the database-level object (with EXCLUDE TABLES) and is individually specified.
- Any name specified in the conditional expression is within the table being specified. (Using table aliases and references to databases, tables, or views that are not specified within the target table result in an error.) It is recommended that the only referenced columns in the conditional expression be the partitioning columns or system-derived column PARTITION of the table. References to other columns does not contribute to partition elimination, and might accidentally qualify more partitions than intended.

LOG WHERE Keyword

If the PARTITIONS WHERE option does not capture all the rows that need to be restored, use the LOG WHERE option. This option inserts into a Teradata-generated error table archived rows that both fall outside the partitions specified by the PARTITIONS WHERE conditional expression *and* match the LOG WHERE conditional expression.

Use the option only if PARTITIONS WHERE is also specified for the object. If LOG WHERE is omitted, the default is to log to the error table only the rows in the partitions being restored that have errors.

ERRORDB/ERRORTABLES Keyword

The ERRORDB and ERRORTABLES options are mutually exclusive; specify only one option for an object. Also, specify either the PARTITIONS WHERE or ALL PARTITIONS option when using either ERRORDB or ERRORTABLES.

- If you specify ERRORTABLES without a database name, or if neither ERRORTABLES nor ERRORDB is specified, the error table is created in the same database as the base table.

- If you do not specify ERRORTABLES, by default the naming convention for the error table is the name of the base table plus the prefix “RS_”. For example, the error table for a table named “DataTable” is “RS_DataTable.” Names are truncated if they exceed 30 bytes.

ALL PARTITIONS Keyword

Use the ALL PARTITIONS option to restore all of the archived partitions in a table. These restrictions apply:

- The object being restored is an individual table, or the ALL FROM ARCHIVE option is specified.
- The source and target tables contain a defined PARTITIONS BY expression.
- The restore is an all-AMP restore rather than a dictionary, cluster, or journal restore.
- PARTITIONS WHERE is not specified for the object.
- The partition bounding condition must have been well-defined when the backup was performed. A bounding condition is well-defined if the PARTITION BY expression on the source table consists of a single RANGE_N function, and if the specified range does not include NO RANGE or UNKNOWN. (Use ANALYZE to determine whether a selected partition is well-defined.)

If a conditional expression is not well-defined, Teradata ARC issues an error, and you must use PARTITIONS WHERE for the restore operation rather than ALL PARTITIONS.

QUALIFIED PARTITIONS Keyword

Use this option only to restore a specific-AMP archive after restoring selected partitions from an all-AMP archive done while an AMP is down.

Examples of Keywords for Restoring Selected Partitions

The next example restores all of the rows of the TransactionHistory table for the month of July 2002:

```
RESTORE DATA TABLES
  (SYSDBA.TransactionHistory)
  (PARTITIONS WHERE
    (! TransactionDate BETWEEN DATE '2002-07-01' AND DATE '2002-07-
31' !))
  ),
RELEASE LOCK,
FILE=ARCHIVE;
```

The following example restores all of the rows for the TransactionHistory table for the month of July 2001 and logs all rows for the month of August 2001 to an error table called TransError:

```
RESTORE DATA TABLES
(SYSDBA.TransactionHistory)
(PARTITIONS WHERE
(! TransactionDate BETWEEN DATE '2001-07-01' AND DATE '2001-07-
31' !),
LOG WHERE
(! TransactionDate BETWEEN DATE '2001-08-01' AND DATE '2001-08-
31' !),
ERRORTABLES SYSDBA.TransError
),
RELEASE LOCK,
FILE=ARCHIVE;
```

The following example restores all data for all tables in database SYSDBA, including all partitions archived for table TransactionHistory:

```
RESTORE DATA TABLES
(SYSDBA)
(EXCLUDE TABLES (TransactionHistory)),
(SYSDBA.TransactionHistory)
(ALL PARTITIONS),
RELEASE LOCK,
FILE=ARCHIVE;
```

Using Keywords with RESTORE

The **RESTORE** statement is affected by the following keywords. For information about the keywords that apply to restoring partitioned data, see [“Restores of Selected Partitions” on page 195](#).

RESTORE DICTIONARY TABLE Keywords

Use the all-AMPs journal archive to restore journal dictionary rows to a Teradata Database that has been reconfigured since the journal archive was created. If you restore the dictionary for an entire database, Teradata ARC restores table, view, macro, stored procedure, and trigger definitions. If you restore specific tables, Teradata ARC restores only the definitions for those tables.

AMP Keyword

Do not specify this option for:

- Dictionary table restores
- Database DBC when you are restoring data tables

With Teradata Database, use the AMP=*n* option to specify up to five AMPs.

This option applies only with the NO FALLBACK TABLE or JOURNAL TABLE options. It is useful following all-AMPs restores when all AMPs are not available at the time of the operation and non-fallback or journal receiving tables are involved.

You must specify the processor to recover with the journal whenever you restore journal tables to specific processors.

If the journal to restore contains after images on a backup processor, the journal rows restored by Teradata ARC are sent to the backup processor for the AMP you specify in the statement.

CLUSTER Keyword

When you specify this option, the following rules apply:

- The option is allowed only when restoring data tables.
- You must restore the dictionary before you restore the cluster.
- You can restore a cluster only from a cluster archive.
- You must perform a build operation on a non-fallback table with unique secondary indexes before you can update it using Teradata SQL.
- The building of fallback data during cluster restores depends on the Teradata Database software release.

Teradata ARC builds all indexes except unique indexes. You cannot update a fallback table with unique indexes using Teradata SQL after a cluster restore until you perform a BUILD operation.

If any AMPs in the list of clusters are offline, you must restore any non-fallback table to those AMPs when they are brought back online. A specific AMP restore can be from a specific AMP or cluster archive.

Teradata ARC releases HUT locks at the cluster level.

RESTORE FALLBACK Keywords

If you do not specify this option, Teradata ARC restores only a single copy and then builds the fallback copy.

If an AMP is down during either the archive or the restore, Teradata ARC does *not* restore non-unique indexes.

If you specify both the RESTORE FALLBACK and NO BUILD options, one of the following occurs:

- If the archive is for all AMPs, then Teradata ARC ignores NO BUILD.
- If the archive and restore are both cluster level, Teradata ARC ignores NO BUILD.
- If the table has unique indexes, the restore operation is not complete until you apply an all-AMPs **BUILD** statement to the fallback tables. Teradata ARC builds non-unique indexes within the cluster.

If you specify both the RESTORE FALLBACK and NO BUILD options to a cluster level archive and the restore is to all AMPs, then Teradata ARC does not build indexes or revalidate the primary subtables. In this case, the table is not fully restored, and you cannot access it using Teradata SQL until you do one of the following:

- Perform another restore operation *without* the NO BUILD option and *with* the RESTORE FALLBACK option.
- Perform another restore operation and apply an all-AMPs **BUILD** statement to the fallback tables.

When you restore a set of cluster archives to a reconfigured Teradata Database, restore each cluster archive to all-AMPs. Use the RESTORE FALLBACK option either on *all* of the cluster restore operations or on *none* of the cluster restore operations. If you do not follow this procedure, and the archive contains fallback tables, Teradata ARC reports an error and terminates the restore operation.

Use NO BUILD for all but the last all-AMPs restore operation. NO BUILD is optional for the last all-AMPs restore operation.

When you restore a set of cluster archives to the same configuration, always perform cluster level restore operations.

If you specify the RESTORE FALLBACK option without the NO BUILD option, the restore operation continues processing on a reconfigured system instead of restarting.

Note: If you are processing a fallback table, this option disables the HALT and PAUSE runtime parameters until the restore completes.

NO BUILD Keywords

These keywords prevent fallback data and secondary indexes from being built on fallback tables when restoring cluster archives to all AMPs. Use this option if you are restoring cluster archives to a Teradata Database that was reconfigured after the cluster archives were created.

When you restore journal tables to a Teradata Database that has been reconfigured since the journal archive was made, specify NO BUILD to prevent sorting of the restored journal and distribution of fallback rows. This procedure is useful if you need to restore more than one journal archive (that is, you need to restore an all-AMPs journal archive and one or more specific AMP journal archives).

If the NO BUILD keywords are used during a **RESTORE** statement, run a separate **BUILD** statement for all databases and/or tables that were restored. The tables will not be accessible until a **BUILD** statement is run.

ALL FROM ARCHIVE Keywords

The ALL FROM ARCHIVE keywords take the place of the database and/or table names that are normally specified after the DATA, DICTIONARY, JOURNAL, or NO FALLBACK TABLES keywords.

You are not allowed to specify any other database or table names to be restored when using ALL FROM ARCHIVE. All databases and tables in the given archive file will be restored, and any existing databases or tables will be overwritten.

CATALOG and Fastpath are not supported while using ALL FROM ARCHIVE. If CATALOG (or Fastpath) is enabled when ALL FROM ARCHIVE is specified, it will be disabled and a warning message will be given.

ALL FROM ARCHIVE cannot be used to restore database DBC, and DBC must be excluded by the user if it is present in the archive being restored.

EXCLUDE TABLES Keyword

Use the EXCLUDE TABLES option to specify one or more tables to skip during a database-level RESTORE. Teradata ARC does not restore any dictionary or data for the specified tables. If any of the tables already exist in the target database, they are unaffected by the RESTORE operation.

Also specify the EXCLUDE TABLES option to retain an existing table during a database-level RESTORE if that table was not archived (or if it was excluded during an ARCHIVE). If you fail to specify the EXCLUDE TABLES option, the table will be dropped and replaced with the data (if any) stored in the archive.

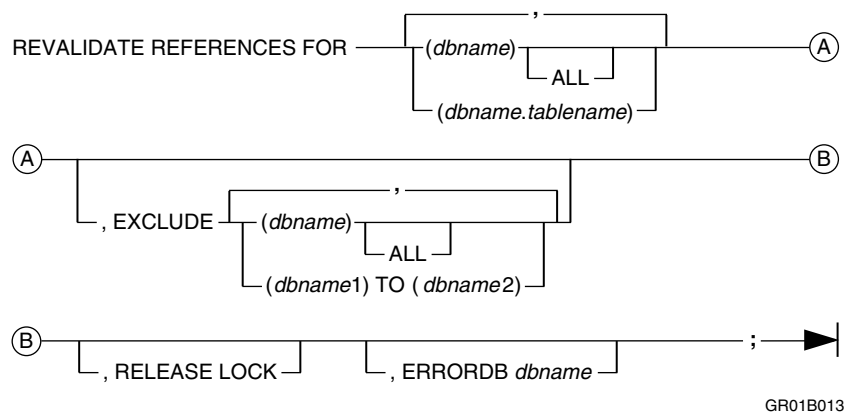
REVALIDATE REFERENCES FOR

Purpose

When a referenced (or parent) or referencing (or child) table is restored, the reference is inconsistently marked in the database dictionary definitions. A table so affected cannot be subject to updates, inserts or deletes.

The **REVALIDATE REFERENCES FOR** statement validates reference indexes that are marked as inconsistent.

Syntax



where

Syntax Element	Description
(dbname)	Name of database to be re-validated.
ALL	Revalidates all child databases of specified database. All database tables with inconsistent indexes are processed.
(dbname.tablename)	Name of a table in the database to be revalidated.
EXCLUDE	Omits identified database from processing.
(dbname)	Name of database excluded from processing.
ALL	Excludes all child databases of specified database.
(dbname1) TO (dbname2)	Alphabetical list of databases to be excluded from processing. Database DBC is NOT included in this list. It will be excluded only if specifically identified.
RELEASE LOCK	Keywords to release HUT lock on table upon completion of processing. (optional)

Syntax Element	Description
ERRORDB	Keyword to generate error tables in designated database. (optional)
(dbname)	Name of database containing error tables generated. If no table name is specified, error tables are created in the database containing the table being processed.

Access Privileges

To revalidate the reference indexes of a table, the user name specified in the **LOGON** statement must have one of the following:

- RESTORE privileges on the table being revalidated
- Ownership privileges on the database or table

Duplicate names in database or tablename lists are ignored. The list of excluded names takes precedence over the list of included names. If a database or table name is covered in both lists, only the exclusion list is effective.

Usage Notes

It is possible to define many referential constraints for a table. When such a table is restored, these constraints are marked *inconsistent*. The **REVALIDATE REFERENCES FOR** statement validates these inconsistent constraints against the target table.

If inconsistent constraints remain after a **REVALIDATE REFERENCES FOR** statement has been executed, the SQL statement **ALTER TABLE DROP INCONSISTENT REFERENCES** must be used to remove them. See [“RESTORE” on page 188](#).

REVALIDATE REFERENCES FOR creates error tables containing information about data rows that failed referential constraint checks.

One error table is created for each referential constraint when the corresponding reference index becomes valid again. The name of each error table consists of the referencing (or child) table name appended by the reference index number. Each error table has the same fields definitions as the corresponding referencing table. The rows it contains are exact copies of the rows that violated the referential constraint in the referencing table.

REVALIDATE REFERENCES FOR performs the following for inconsistent reference indexes that can be validated:

- Validates the inconsistent reference index on the target table and its parent/child tables
- Creates an error table
- Inserts rows that fail the referential constraint specified by the reference index into the error table

If an inconsistent reference cannot be validated, the index is skipped and reported in a message at the end of the operation.

For information on referential integrity, see the *SQL Reference: Fundamentals*.

Example

Suppose an Employee table has just been restored with two inconsistent references defined on the table. One indicates the Employee table references Department table; the other indicates Employee table references Project table.

Now, Project table is dropped before a restore operation and the following statement is submitted:

```
REVALIDATE REFERENCES FOR (Employee);
```

Assume the referential constraint specifying Employee references Department has reference index number 4. The following occurs:

- 1 Error table Employee_4 is created with information about the data rows in Employee that reference Department.
- 2 The reference index specifying Employee table is referencing Department table is validated.
- 3 The reference index specifying that Employee table is referencing Project table remains inconsistent. A message reports that Employee table still contains inconsistent reference indexes.
- 4 Submit the SQL statement:

```
ALTER TABLE EMPLOYEE DROP INCONSISTENT REFERENCES;
```

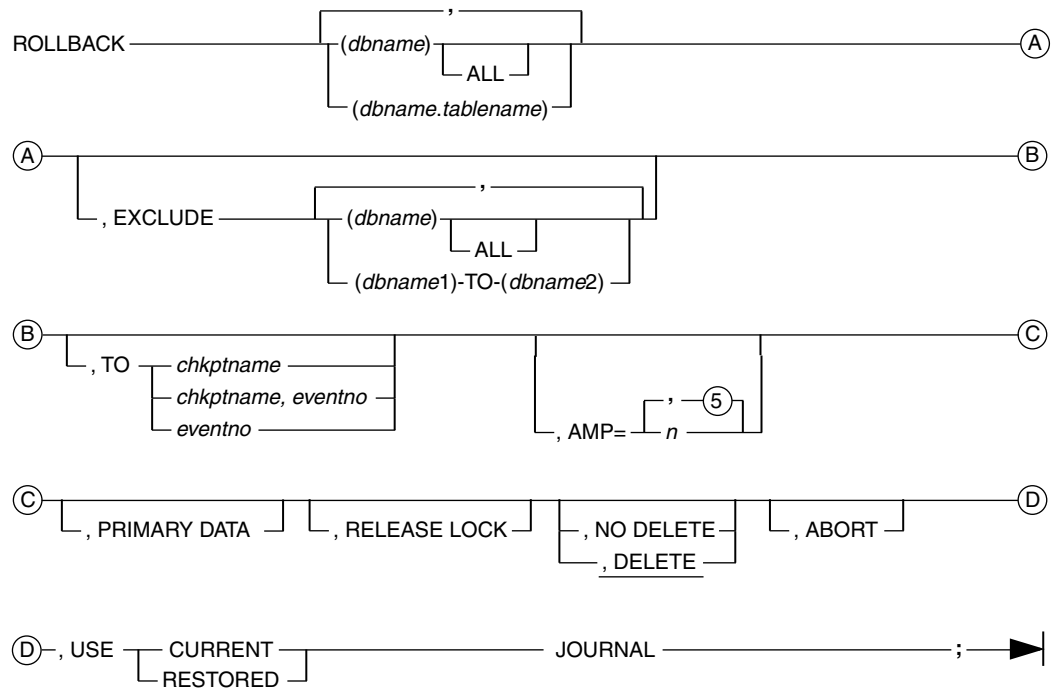
- 5 Then, query error table Employee_4 to correct the data rows in the Employee table.

ROLLBACK

Purpose

The **ROLLBACK** statement recovers specified data tables, with before journals, to the state they were in *before* they were modified.

Syntax



2412B014

where

Syntax Element	Description
<i>(dbname)</i>	Name of the database to be rolled back.
ALL	Indicates that the rollback affects the named database and its descendants. ALL recovers all database tables with after images.
<i>(dbname.tablename)</i>	Name of the table. Specify only distinct tables within the list. If you are recovering a specific AMP, the specified table must be non-fallback because fallback tables are completely recovered by any previous all-AMPs rollback.

Syntax Element	Description
EXCLUDE	Prevents the named database from being recovered.
<i>(dbname)</i>	Name of the excluded database.
ALL	Excludes the named database and its descendants from being recovered.
<i>(dbname1) TO (dbname2)</i>	Alphabetical list of client databases to be excluded from the recovery operation. The delimiting database names need not identify actual databases. Database DBC is not included as part of a range.
<i>chkptnamechkptname, eventno or eventno</i>	Specifies the termination point for the rollback.
PRIMARY DATA	Applies primary and fallback row images during the rollback process. Teradata ARC ignores secondary index rows and fallback rows for online AMPs.
RELEASE LOCK	Releases utility locks on the named databases automatically when the rollback operation completes.
NO DELETE or DELETE	Deletes, or refrains from deleting, the restored journal subtable after the rollback is complete. The default is DELETE.
ABORT	Stops the operation if an AMP to which a non-fallback archive is to be recovered is offline. This option does not affect specific AMP operations.
USE CURRENT JOURNAL or USE RESTORED JOURNAL	Uses the current journal or a subtable that was previously restored to the Teradata Database. CURRENT uses the active journal subtable followed by any saved subtable.

Access Privileges

To rollback a database or a table in the database, the user name specified in the **LOGON** statement must have one of the following:

- The RESTORE privilege on the objects to recover
- Ownership of the objects to recover

Usage Notes

The **ROLLBACK** statement erases all changes made to a database or selected tables of a database. The **ROLLBACK** statement can be used to roll back data tables on all or a subset of the processors. A rollback uses a single journal table as input. All objects in the object list must share the same journal table.

A rollback is not allowed over a data definition statement that changed a table's structure. If Teradata ARC encounters such a change to a table during a rollback, the rollback for that table stops and the program prints a message in the output listing.

If you are recovering a nonfallback table with unique secondary indexes and an AMP is offline, Teradata ARC invalidates unique secondary indexes on the table. After the AMP returns online, drop and recreate the unique secondary indexes for that table.

You can use the **RELEASE LOCK** statement instead of the **RELEASE LOCK** keywords. Use the **RELEASE LOCK** *statement* to release all HUT locks and the **RELEASE LOCK** *keyword* to release specific HUT locks.

To roll forward using a current single image journal, first roll back to the checkpoint. After the rollback has completed, use **ROLLFORWARD** to restore the data.

chkptname Parameter

Use the **CHECKPOINT** statement to generate the checkpoint for the journal you use in the recovery (the one for the data tables you want to roll back).

If you specify this option, Teradata ARC scans the journal table for the identified checkpoint. If that checkpoint does not exist, it returns an error.

If you use an unqualified *chkptname* and there are multiple rows in the journal table that all have the same name, Teradata ARC uses the *last* chronological entry made. In the **ROLLFORWARD** statement, on the other hand, Teradata ARC uses the *first* chronological entry made.

Alpha characters in a *chkptname* are not case sensitive. For example, *chkptname* ABC is equal to *chkptname* abc. If you do not specify this option, Teradata ARC uses the entire journal table.

DELETE Keyword

Do not specify any **DELETE** options when you use the current journal for the recovery. Use **NO DELETE** when you want to recover selected tables and then later want to recover the balance of the tables that might have changes in the journal.

ABORT Keyword

ABORT aborts an all-AMPs roll operation that includes non-fallback tables or single copy journal tables if an AMP is down. This option does not affect specific AMP roll operations.

PRIMARY DATA Keywords

Use this keyword phrase to reduce the amount of I/O during rollback. Using this option improves the rollback performance when you are recovering a specific AMP from a disk failure.

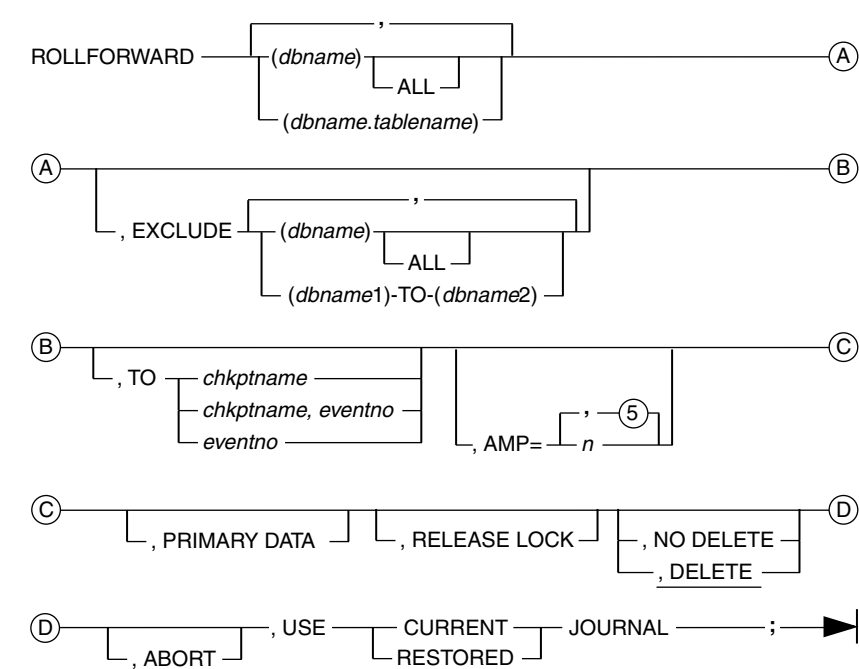
If you are recovering a specific AMP, unique indexes are invalid, so always follow a **ROLLBACK** statement that uses the **PRIMARY DATA** option with a **BUILD** statement.

ROLLFORWARD

Purpose

The **ROLLFORWARD** statement recovers specified data tables, with after journals, to their state following a modification.

Syntax



2412A015

where

Syntax Element	Description
(dbname)	Name of the database to be rolled forward.
ALL	Rolls forward the named database and its descendants.
(dbname.tablename)	Name of the table to be rolled forward. If you are recovering a specific AMP, the specified table must be non-fallback because fallback tables are completely recovered by any previous activity.
EXCLUDE	Prevents the named database from recovery.
(dbname)	Name of the database excluded from recovery.

Syntax Element	Description
<i>ALL</i>	Excludes from recovery the named database and its descendants.
<i>(dbname1) TO (dbname2)</i>	Alphabetical list of client databases to be excluded from recovery. The delimiting database names need not identify actual databases. Database DBC is not included as part of a range.
<i>chkptname, chkptname, eventno or eventno</i>	Specifies the termination point for the rollforward operation.
PRIMARY DATA	Applies primary and fallback row images during the rollforward process. Teradata ARC ignores secondary index rows and fallback rows for online AMPs.
RELEASE LOCK	Releases utility (HUT) locks on the named databases automatically when rollforward completes.
NO DELETE or DELETE	Deletes, or refrains from deleting, the restored journal subtable after the rollforward is complete. The default is DELETE.
ABORT	Aborts the roll operation if an AMP to which a non-fallback archive to be recovered is offline. This option affects only all-AMPs roll operations. It does not affect specific AMP operations.
CURRENT or RESTORED	Uses the current journal or a subtable that was previously restored to the Teradata Database. CURRENT uses the active journal subtable followed by any saved subtable.

Access Privileges

To roll forward a database or a table in the database the user name specified in the **LOGON** statement must have one of the following:

- The RESTORE privilege on the objects being recovered
- Ownership of the objects

Usage Notes

A rollforward operation uses a single journal table as its input. All tables you specify for recovery must be defined with the same journal table. A **ROLLFORWARD** statement can recover either all data tables or a subset of the tables that are protected by the same journal table.

Use the **ROLLFORWARD** statement to rollforward data tables on all or a subset of the AMPs.

You cannot execute a rollforward over a data definition statement that changed a table's structure. If Teradata ARC encounters such a change to a table during a rollforward, the rollforward for that table stops and an error message is placed in the output listing.

If you are recovering a nonfallback table with unique secondary indexes and an AMP is offline, then Teradata ARC invalidates unique secondary indexes on the table. After the AMP returns online, you must drop and then recreate the unique secondary indexes for that table or use the **BUILD DATA TABLES** statement on the table.

The **BUILD DATA TABLES** statement is *much* faster than dropping and recreating the unique secondary indexes.

If the table being recovered has a single after image journal and the recovery operation uses the current journal, then you cannot roll forward data rows on the AMP that are backed up by an AMP that is offline. If the table being recovered has a local single after image journal, it may be recovered only to the last archived data.

If the tables to be rolled forward have a dual after-image journal, then the rollforward operation (by default) uses the journal tables that exist on each AMP to roll forward those AMPs.

When you perform a restore and rollforward, you can bring the affected data tables on the AMPs to their state after a modification. Specific AMP rollforwards are typically performed to do one of the following:

- Complete a previous all-AMPs rollforward that happened while some AMPs were offline.
- Recover a single AMP from a disk failure.

chkptname Parameter

Use the **CHECKPOINT** statement to generate the checkpoint entry for the journal you use in the recovery (the one for the data tables you want to roll forward).

If you specify this option, Teradata ARC first scans the journal table for the identified checkpoint. If the checkpoint does not exist, Teradata ARC returns an error.

If you use an unqualified *chkptname*, and there are multiple checkpoint rows in the journal table with the same name, Teradata ARC uses the *first* chronological entry made. Note this difference from the **ROLLBACK** statement, where Teradata ARC uses the *last* entry made.

Alpha characters in a *chkptname* are not case sensitive; for example, *chkptname* ABC is equal to *chkptname* abc. If you do not specify this option, Teradata ARC uses the entire journal table.

PRIMARY DATA Keywords

Use this keyword phrase to reduce the amount of I/O during rollforward. Using this option improves the rollforward performance when recovering a specific AMP from a disk failure.

If you are recovering a specific AMP, unique indexes are invalid, so always follow a **ROLLFORWARD** statement that uses the PRIMARY DATA option with a **BUILD** statement.

RELEASE LOCK Keywords

With Teradata Database, RELEASE LOCK does not release HUT locks on non-fallback tables with remote single after image journaling when the backup AMP is offline.

DELETE Keyword

Use NO DELETE keyword phrase to recover selected tables if you will later want to recover the balance of the tables with changes in the journal.

Do not specify any DELETE options when you use the current journal for the recovery.

ABORT Keyword

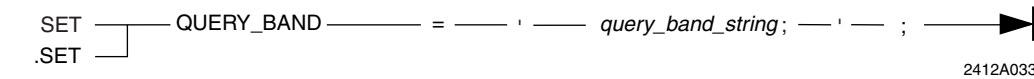
The ABORT keyword aborts an all-AMPs roll operation that includes non-fallback tables or single copy journal tables if an AMP is down. This option does not affect specific AMP roll operations.

SET QUERY_BAND

Purpose

The **SET QUERY_BAND** statement provides values for a session to precisely identify the origins of a query.

Syntax



where

Syntax Element	Description
query_band_string	Associates a name with a value. Example SET QUERY_BAND = 'Job=payroll;' ;

Usage Notes

A query band is a set of name and value pairs that can be set on a session to identify where a query originated. These identifiers are in addition to the current set of session identification fields, such as *user id*, *account string*, *client id*, and *application name*. Query bands offer a way to identify the user and application, for example, when SQL-generating tools and web applications use pooling mechanisms that hide the identity of users because each connection in the pool logs into the database using the same user account. Without query bands, there is no way to tell the source of the request when the request comes from a multi-tiered application. Another potential use of query bands is troubleshooting, when it is necessary to provide the specific user, application, or report that issued a request.

All **SET QUERY_BAND** statements submitted by Teradata ARC are session query bands. A session query band is stored in a session table and recovered after a system reset. (Teradata ARC does not use transaction query bands.)

Restarting Teradata ARC

This chapter describes how to recover if your client system, Teradata ARC, or the Teradata Database fail during an archive/recovery operation. Topics include:

- [Restart Log](#)
- [Restarting Teradata ARC](#)
- [Restart After Client or Teradata ARC Failure](#)
- [Restart After a Teradata Database Failure](#)
- [Recovery Control Catalog](#)

Restart Log

Teradata ARC does not record most of the modifications it makes to a database in the transient journal. Instead, Teradata ARC uses a file called *restart log* to maintain the operation status of the job being run.

The restart log catalogs the effects of utility operations on a database or table. Teradata ARC writes restart entries in the log at the beginning and end of each significant step (and at intermediate points in a long running task). If a job is interrupted for some reason, Teradata ARC uses this log to restart the archive or restore operation that was being performed.

When you start Teradata ARC, it first places markers into the restart log to keep track of the current input and logon statements and then copies the command stream to the restart log. Teradata ARC uses these commands during restart to continue with the most recent activity using the same logon user originally entered.

Restarting Teradata ARC

The Teradata Database uses task execution logic to restart an archive or to restore jobs. The logic differs depending on the operation, as follows.

During a Database DBC Operation

If a client system or Teradata ARC failure interrupts an archive or restore of database DBC, you cannot restart the operation. In this case, perform the entire restore process again, including re-initializing the Teradata Database and the dictionary tables.

Use the RESTART option if a multiple database archive or restore that includes database DBC is interrupted during the archive or restore of some database other than database DBC. But before you restore database DBC, initialize the Teradata Database by doing the following:

- 1 Reconfigure the system from a single AMP to the desired configuration.
- 2 Run the DBC Initialization Procedure (DIP) to initialize system views, macros, triggers and users, and the error messages tables. Refer to the software release cover letter for information on running DIP.

During an Archive Operation

Restarting an archive operation causes Teradata ARC to:

- Reposition the output archive file at the data block indicated in the last restart point recorded in the restart log before the failure occurred.
- Restart the operation at the beginning of the last table or secondary index subtable that was being archived before the failure. Teradata ARC performs this step only if the AMP configuration has changed, that is, if any of the AMPs are online before the failure but online after the failure.

In the case of an interrupted archive of database DBC, resubmit the **ARCHIVE** statement without specifying the RESTART parameter.

During a Restore Operation

Restarting a restore operation causes Teradata ARC to:

- Reposition the input archive file at the data block indicated in the last restart point recorded in the restart log before the failure occurred.
- Restart the operation at the beginning of the last table or secondary index subtable that was being restored before the failure. Teradata ARC performs this step only if the AMP configuration has changed, that is, if any of the AMPs are online before the failure but online after the failure.

During a Recovery Operation

Restarting a recovery (rollback or rollforward) operation causes Teradata ARC to:

- Resubmit the original recovery request to the Teradata Database.
- Reinitiate the complete recovery action.

The Teradata Database reads the entire input journal subtable and ignores recovery actions for change images already used.

Restart After Client or Teradata ARC Failure

If your client system or Teradata ARC fails during an archive, restore, or recovery procedure, you can restart the operation. Teradata ARC responds to the failure by logging all its sessions

off the Teradata Database. Because the Teradata Database does not journal Teradata ARC statements, Teradata ARC takes no recovery action at logoff. Database locks placed by a utility statement on databases or tables remain intact following a failure.

You can restart a utility operation automatically if you specify the RESTART runtime option when you invoke Teradata ARC. It uses the restart log to determine the state of the pending operation. Command execution continues using the copy of the original source statement file. Teradata ARC reads the restart log to determine its restart point, then reenters the statement that was being processed at the time of the failure.

If a client or utility failure interrupted an archive or restore of database DBC, you cannot restart the operation.

If an archive or restore of multiple databases including database DBC is interrupted, you must resubmit the **ARCHIVE** statement, but you can also specify the RESTART option.

Restarting an Archive Operation

Restarting an archive operation causes Teradata ARC to:

- Reposition the output archive file at the data block indicated in the last restart point recorded in the restart log before the failure occurred.
- If the AMP configuration changed (AMPs were online before the failure but offline afterward), restart begins with the last table being archived before the failure.

Examples

The following examples show the restart of an all-AMPs archive on VM and MVS.

Note: The RESTART parameter has been added to the EXEC.

Restarting an Archive on VM

```
/* Minimal EXEC to run ARCMAN */
address command
'GLOBAL LOADLIB LSCRTL'
'GLOBAL TXTLIB CLI'
'SET LDRTBLS 10'
'CP SET TIMER REAL'
'FILEDEF DBCLOG DISK DBCLOG LOGFILE A (RECFM F LRECL 32760'
'FILEDEF DUMP100 TAP1 SL (BLKSIZE 32760 RECFM U DEN 6250'
'LABELDEF DUMP100 VOLID 000100 VOLSEQ 0001 SEC 0'
'ARCMAN <ARC.CNTL.A RESTART'
```

The SYSIN control file, ARC CNTL on the CMS A-disk, contains the following:

```
LOGON DBC,DBC;
DUMP DATA TABLES (DBC) ALL,
RELEASE LOCK,
INDEXES,
FILE=DUMP100;
LOGOFF;
```

The files ARCMAN MODULE, CLI TXTLIB, and LSCRTL LOADLIB must be on minidisks or SFS directories that are ACCESS'ed from the CMS virtual machine. The latter two files are the CLILv2 and SAS/C runtime libraries, respectively.

Restarting an Archive on MVS

```
//DBC DMP1 JOB 1,'DBC OPERATIONS',REGION=2048K,MSGCLASS=A //DUMP
EXEC PGM=ARCMAN,PARM='RESTART'
//STEPLIB DD DSN=DBC.AUTHLOAD,DISP=SHR
// DD DSN=DBC.TRLOAD,DISP=SHR
//DBCLOG DD DSN=DBC.ARCLOG.DATA,DISP=OLD
//DUMP100 DD DISP=OLD,DSN=DBC.DUMP100
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,BLKSIZE=132,LRECL=132)
//SYSUDUMP DD SYSOUT=*
//SYSIN DD DATA,DLM=##
LOGON DBC,DBC;
ARCHIVE DATA TABLES (DBC) ALL,
    RELEASE LOCK,
    INDEXES,
    FILE=DUMP100;
LOGOFF;
##
```

In this example, the tape with VOLSER=DGJA is mounted at the time that Teradata ARC abends.

Note: When archiving on tape for MVS, specify the tape level information and all tape volume serial numbers, including the one mounted at the time of the abend.

Restarting a Restore Operation

Restarting a restore operation causes Teradata ARC to:

- Reposition the input archive file at the data block indicated in the last restart point recorded in the restart log before the failure occurred.
- If the AMP configuration changed (AMPs are online before the failure but offline afterward), restart begins with the last table being restored before the failure.

If a restore of database DBC is interrupted for any reason, perform the entire restoration again, including re-initializing the Teradata Database. Even if the Teradata Database was initialized prior to database DBC restore, you must initialize it again.

If archive of database DBC is interrupted, resubmit the **ARCHIVE** statement without specifying the **RESTART** option.

Examples

- For VM, add the **RESTART** parameter on the command line, which invokes ARCMAN. The SYSIN control file is read from the restart log on restart.
- For MVS, the only difference between the original job and the restarted job is to add the **RESTART** parameter to the **PARM** statement of the EXEC.

Restarting a Restore on VM

```
/* Minimal EXEC to run ARCMAN */
trace off
address cms
"GLOBAL LOADLIB LSCRTL"
"GLOBAL TXTLIB CLI"
"CP SET TIMER REAL"
"FILEDEF DBCLOG DISK DBCLOG LOGFILE A (RECFM F LRECL 32760"
```



```
"FILEDEF DUMP100 TAP1 SL (BLKSIZE 32760 RECFM U DEN 6250"
"LABELDEF DUMP100 VOLID 000100 VOLSEQ 0001 SEC 0"
"ARCMAN <ARC.CNTL RESTART"
```

The SYSIN control data set, ARC CNTL A, contains the following:

```
LOGON DBC, DBC;
RESTORE DATA TABLES (PERSONNEL) ALL,
    RELEASE LOCK,
    FILE=DUMP100;
LOGOFF;
```

Restarting a Restore on MVS

```
//DBCRST1 JOB 1, 'DBC OPERATIONS', REGION=2048K, MSGCLASS=A
//RESTORE EXEC PGM=ARCMAN, PARM='RESTART'
//STEPLIB DD DSN=DBC.AUTHLOAD, DISP=SHR
//          DD DSN=DBC.TRLOAD, DISP=SHR
//DBCLOG DD DSN=DBC.ARCLOG.DATA, DISP=OLD
//DUMP100 DD DSN=DBC.DUMP100, DISP=OLD
//SYSPRINT DD SYSOUT=*, DCB=(RECFM=F, BLKSIZE=132, LRECL=132)
//SYSIN DD DATA, DLM=##
LOGON DBC, DBC;
RESTORE DATA TABLES (PERSONNEL) ALL,
    RELEASE LOCK,
    FILE=DUMP100;
LOGOFF;
##
```

Restarting a Checkpoint Operation

When the **CHECKPOINT** statement writes rows to a journal, the transient journal also keeps a record of these rows. Consequently, if the client or Teradata ARC fails, the results of the checkpoint operation are rolled back by the transient journal.

- If a Teradata ARC statement specified checkpoints on multiple journal tables, Teradata ARC rolls back only the checkpoint being taken at the time of the client or utility failure.
- If a **CHECKPOINT** statement terminates abnormally, all locks placed by Teradata ARC are removed.

When Teradata ARC restarts, the **CHECKPOINT** statement completes.

Restart After a Teradata Database Failure

If the Teradata Database fails during an archive, restore or recovery procedure, you can restart the operation. After a failure and restart, sessions remain intact and Teradata ARC continues processing with the statement that was being executed at the time of the failure.

Restarting an Archive Operation

Restarting an archive operation causes Teradata ARC to:

- Reposition the output archive file at the data block indicated in the last restart point recorded in the restart log before the failure occurred.

- If the AMP configuration changed (AMPs are online before the failure but offline afterward), restart begins with the last table being archived before the failure.

If the Teradata Database fails during an archive of database DBC, Teradata ARC's default action is to wait for the Teradata Database to come back online and then try the archive operation again. Some options (for example, ABORT, HALT, and PAUSE) can affect whether Teradata ARC stops, but the default is to try again after the system recovers.

Restarting a Restore Operation

Restarting an restore operation causes Teradata ARC to:

- Reposition the input archive file at the data block indicated in the last restart point recorded in the restart log before the failure occurred.
- If the AMP configuration changed (AMPs are online before the failure but offline afterward), restart begins with the last table being archived before the failure.

If the failure occurred during a restore of database DBC, repeat the entire restore operation, including re-initializing the Teradata Database. Some options (for example, ABORT, HALT, and PAUSE) can affect whether Teradata ARC stops, but the default is to try again after the system recovers.

Restarting a Recovery Operation

Restarting a recovery (rollback or rollforward) operation causes Teradata ARC to:

- Resubmit the original recovery request to the Teradata Database.
- Reinitiate the complete recovery action.

The Teradata Database reads the entire input journal subtable, ignoring recovery actions for change images already processed.

Restarting a Checkpoint Operation

When the **CHECKPOINT** statement writes rows to a journal, the transient journal also keeps a record of these rows. Consequently, if the Teradata Database fails, the results of the checkpoint operation are rolled back by the transient journal.

- If Teradata ARC statement specified checkpoints on multiple journal tables, Teradata ARC rolls back only the checkpoint being taken at the time of the hardware failure.
- If a **CHECKPOINT** statement terminates abnormally, all locks Teradata ARC placed are removed.

When Teradata ARC restarts, the **CHECKPOINT** statement completes.

Removing HUT Locks After a Restart

Teradata ARC places HUT locks on tables and databases to prevent modifications to the tables and databases before you can restart the job. consequently, when a Teradata ARC operation does not finish, HUT locks might remain on tables and databases.

Occasionally, you might decide that you cannot restart a job for some reason. In this case, you can specifically remove HUT locks to allow the database or table to be accessed again. To remove HUT locks:

- 1 Run a utility statement that specifies the **RELEASE LOCK** statement for the currently locked database.
- 2 Specify a new restart log. (The restart log used when the failure occurred indicates that a restart is necessary, which prevent the removal of Teradata ARC HUT locks.)

Example

The following example shows the statements needed to release the locks on a database named *Database_Name*.

```
LOGON TDPI/DBC,DBC;
RELEASE LOCK ( Database_Name ),  OVERRIDE;
LOGOFF;
```

For additional information on the **RELEASE LOCK** statement and the options available on this statement, refer to [“RELEASE LOCK” on page 185](#).

Recovery Control Catalog

The Teradata Database maintains a Recovery Control Catalog (RCC) to record all archive and recovery activity. Use the RCC to monitor restart and recovery activity, or use the RCVMANAGER utility.

The RCC consists of the following standard tables in the database DBC:

- *RCEvent* contains a row for each archive and/or recovery activity.
- *RCConfiguration* contains a row for each archive and/or recovery activity that did not include all AMPs defined in the hardware configuration.
- *RCMedia* contains a row for each removable device used in either an archive or recovery activity.

User DBC must authorize access to these tables for selection and deletion operations. The Teradata Database automatically generates all inserts to the tables, so remove unnecessary rows from the table.

Recording Row Activity

Each row in the *RCEvent* table contains multiple columns. Some of these columns are the same for all rows and others are optional, depending on the type of event being recorded.

Standard Fields

- **Event Number** - A unique integer value assigned by the Teradata Database to each recorded event.

Teradata ARC assigns event numbers in ascending order. For example, events that occur earlier in time have smaller numbers. The event number (*not* the date and time stamp) is the most accurate way to determine the chronology of events.

- **Date and Time** - The value of the Teradata Database time-of-day clock when the event is recorded.

Because earlier events can have later time stamps than later events, *do not* use this value to determine the chronology of checkpoint events. Instead, use event numbers.

- **User Name** - Logon ID for the user who initiated the event.
- **Event Type** - A text string identifying the event. The event types are:

- | | |
|------------------|------------------|
| • BUILD | • DUMP (ARCHIVE) |
| • CHECKPOINT | • RESTORE |
| • COPY | • ROLLBACK |
| • DELETE JOURNAL | • ROLLFORWARD |

- **Database Name** - The name of the database affected by the activity. If multiple databases are affected, a row is recorded for each database.

- **Object Type** - A single character that defines the type of object involved. The character values and their meanings are:

- D - total database
- T - data table
- J - journal table

- **Object Identifier** - A four byte field containing either a database ID or the uniqueness portion of a table identifier.

This field further defines the object involved in the event. These fields are derived from the DBASE and TVM dictionary tables and are maintained by the Teradata Database.

- **All-AMPs Flag** - The value "A" means the following:

- All AMPs
- Rows in RCConfiguration define any offline AMPs
- Rows in RCConfiguration define the list of AMPs involved
- Rows in RCConfiguration define the list of AMPs involved

- **Restart Count** - If the event is restarted and the online AMP configuration has changed, this column increases by one.

If the event was completed without a configuration change, this field contains a zero.

- **Operation in Progress** - This field contains a Y when the Teradata ARC operation specified is not complete.

Operations that terminate abnormally and are not restarted are left with Y. When the specified Teradata ARC operation completes, this field contains an N.

Optional Fields

The following are optional fields in the event rows:

- **Dataset Name** - The client file name specified for an archive or recovery activity.

- **Table Name** - The table name specified for an archive or recovery activity.
Contains one row for each table affected unless all tables are affected. If all rows are affected, BTEQ shows a question mark. This indicates a NULL field.
- **Checkpoint Name** - The label you give to the checkpoint.
- **Linking Event Number** - The number assigned to some other event that influenced the current event.
The termination point for a rollback is a linking event number.
- **Journal Used** - A single character indicating which journal table the event used:
 - C - current journal table
 - R - restored portion of the journal table
 - S - saved portion of the journal table
- **Journal Saved** - Y or N to indicate that the **CHECKPOINT** statement did (or did not) contain the SAVE option.
- **Index Present** - Y or N to indicate whether the archive included the index option.
- **Dup Archive Set** - Y or N to indicate if this is a duplicate file created by an archive request.
If two files are created by a single archive request, two event rows are generated. One has the flag set to Y to indicate that it is a duplicate. The other has a value of N to indicate that it is the primary copy.
- **Lock Mode** - A single character to identify the HUT lock that is applied by an archive operation:
 - A - The operation executed under either an access lock or a group read lock.
 - R - The operation executed under a read lock.

The *RCEvent* table also has other fields not listed above, which are reserved for future use.

Recording AMP Information

Teradata ARC inserts rows into the *RCConfiguration* table for each archive activity that does not affect all of the AMPs in the configuration:

- If the activity is for all of the AMPs, but some AMPs are offline, Teradata ARC records a row for each offline AMP.
- If the activity is for specific AMPs, Teradata ARC records a row for each AMP that is both specified and online.

A row contains these attributes:

- The event number assigned to the activity.
- The logical identifier for the processor.
- The cabinet and slot number for the processor.
- A state flag of D (down) to indicate that a processor was offline during an all-AMPs operation, or a flag of U to indicate the processor participated in a specific AMPs operation.
- A restart count to indicate during which restart of the event the row was inserted.

If the Teradata Database fails during an archive or recovery activity and the operation is restarted *and* the configuration changes, then Teradata ARC writes rows to the *RCEvent* and *RCConfiguration* tables. The restart count ties these rows together.

Recording Device Information

The utilities insert rows into the *RCMedia* table for each removable device used in an archive or recovery activity. A row contains these attributes:

- The event number assigned to the activity.
- The six-character volume serial assigned to the removable device.
- A sequence number that assigns the device its position within the set.
- A duplicate-file flag to indicate the device belongs to a duplicate file created by an archive request. Y indicates a duplicate; N indicates the primary file.

How to Read Syntax Diagrams

This appendix describes the conventions that apply to reading the syntax diagrams used in this book.

Syntax Diagram Conventions

Notation Conventions

The following table defines the notation used in this section:

Item	Definition / Comments
Letter	An uppercase or lowercase alphabetic character ranging from A through Z.
Number	A digit ranging from 0 through 9.
	Do not use commas when entering a number with more than three digits.

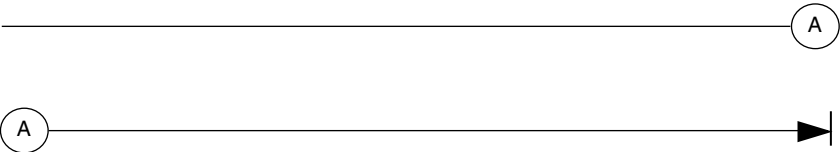
Item	Definition / Comments	
Word	Variables and reserved words.	
	IF a word is shown in...	THEN it represents...
	UPPERCASE LETTERS	a keyword.
		Syntax diagrams show all keywords in uppercase, unless operating system restrictions require them to be in lowercase.
		If a keyword is shown in uppercase, you may enter it in uppercase or mixed case.
	lowercase letters	a keyword that you must enter in lowercase, such as a UNIX command.
	lowercase italic letters	a variable such as a column or table name.
		You must substitute a proper value.
	lowercase bold letters	a variable that is defined immediately following the diagram that contains it.
	UNDERLINED LETTERS	the default value.
		This applies both to uppercase and to lowercase words.
Spaces	Use one space between items, such as keywords or variables.	
Punctuation	Enter all punctuation exactly as it appears in the diagram.	

Paths

The main path along the syntax diagram begins at the left, and proceeds, left to right, to the vertical bar, which marks the end of the diagram. Paths that do not have an arrow or a vertical bar only show portions of the syntax.

The only part of a path that reads from right to left is a loop.

Paths that are too long for one line use continuation links. Continuation links are small circles with letters indicating the beginning and end of a link:



FE0CA002

When you see a circled letter in a syntax diagram, go to the corresponding circled letter and continue.

Required Items

Required items appear on the main path:

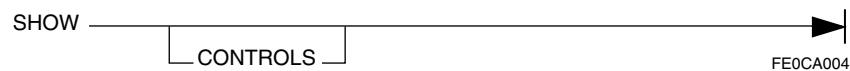


If you can choose from more than one item, the choices appear vertically, in a stack. The first item appears on the main path:

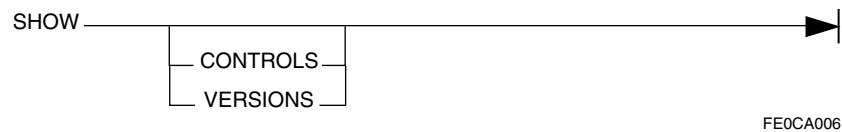


Optional Items

Optional items appear below the main path:



If choosing one of the items is optional, all the choices appear below the main path:



You can choose one of the options, or you can disregard all of the options.

Abbreviations

If a keyword or a reserved word has a valid abbreviation, the unabbreviated form always appears on the main path. The shortest valid abbreviation appears beneath.

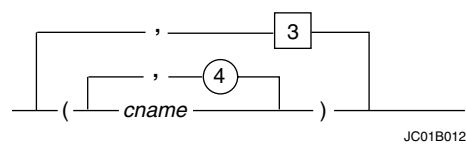


In the above syntax, the following formats are valid:

- SHOW CONTROLS
- SHOW CONTROL

Loops

A loop is an entry or a group of entries that you can repeat one or more times. Syntax diagrams show loops as a return path above the main path, over the item or items that you can repeat.



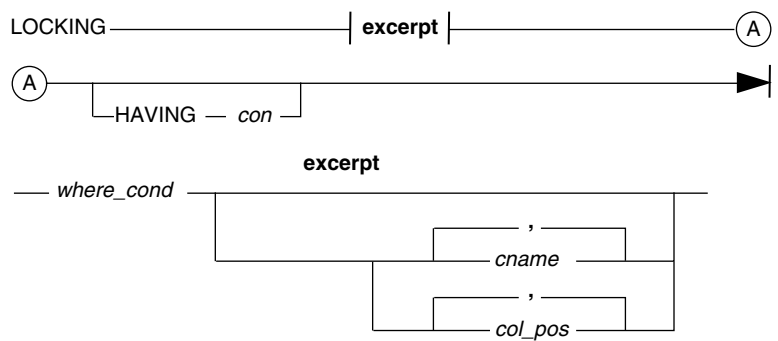
The following rules apply to loops:

If...	Then...
there is a maximum number of entries allowed	<p>the number appears in a circle on the return path.</p> <p>In the example, you may enter <i>cname</i> a maximum of 4 times.</p>
there is a minimum number of entries required	<p>the number appears in a square on the return path.</p> <p>In the example, you must enter at least 3 groups of column names.</p>
a separator character is required between entries	<p>the character appears on the return path.</p> <p>If the diagram does not show a separator character, use one blank space.</p> <p>In the example, the separator character is a comma.</p>
a delimiter character is required around entries	<p>the beginning and end characters appear outside the return path.</p> <p>Generally, a space is not needed between delimiter characters and entries.</p> <p>In the example, the delimiter characters are the left and right parentheses.</p>

Excerpts

Sometimes a piece of a syntax phrase is too large to fit into the diagram. Such a phrase is indicated by a break in the path, marked by | terminators on either side of the break. A name for the excerpted piece appears between the break marks in boldface type.

The named phrase appears immediately after the complete diagram, as illustrated by the following example.



JC01A014

APPENDIX B

Multivolume CMS Tape Support

Conversational Monitor System (CMS) tape messages are routed to the host system operator.

CMS Tape Support Messages

To support multivolume tape support under CMS, messages are generated when Teradata ARC runs under CMS with tape input or output.

Messages are divided into two groups:

- Messages routed to the host system operator
- Messages routed to the console of the virtual machine that is running Teradata ARC

Table 16 gives a brief description of each message that is routed to the host system operator.

Table 16: Messages Routed to System Operator

Message	Description
DDNAME: XXXXXXXX, BAD FILE SEQUENCE XXX, MUST BE 1	An invalid file sequence number is specified when file sequence number 1 was expected.
DDNAME: XXXXXXXX NOT DEFINED	The ddname in the LABELDEF statement is not defined in the FILEDEF statement.
EXPIRATION DATE NOT EXPIRED	The expiration date in the LABELDEF is not reached.
FIRST VOLSER INVALID: XXXXXX, SHOULD BE XXXXXX	The incorrect volume 1 tape is mounted. The message also displays the volume ID of the tape that is mounted and the volume ID of the tape that should be mounted.
MISSING LABELDEF FOR SL TAPE FILE: XXXXXXXX	The LABELDEF for the ddname is missing.
PLEASE MOUNT VOL XXXXXX ON XXX, XXX, RING XXX FOR DDNAME XXXXXXXX VOLSEQ XXX	The program is ready for the first tape in the dump or RESTORE operation.
RECORD XXXXXX NOT FOUND ON VOL XXXXXX SEQ XXX	The requested record is not found.
TAP1 (181) IS NOT DEFINED, ATTACH TAPE DRIVE AND RETRY	No device is defined for address x'181'. This message is routed to the console of the virtual machine that is running Teradata ARC.

Table 16: Messages Routed to System Operator (continued)

Message	Description
TAPE MOUNTED IS NL--SL REQUESTED	A tape without a label is mounted but a standard label is requested.
TAPE MOUNTED IS SL--NL REQUESTED	A standard label tape is mounted but a tape without a label is requested.
TAPE ON XXX IS FILE PROTECTED	An attempt is made to write on a tape that is write-protected.
TAPE ON XXX IS XXXXXX, XXX, SEQ XXX	The tape that is mounted is the correct tape.
TAPE: XXXXXX, WRONG DSNAME: XXXXXXXXXXXXXXXXXXXX	The incorrect data set name is being read for the specified volume.
VOLUME: XXXXXX HDR1 LABEL MISSING OR INVALID	HDR1 information in the LABELDEF statement is missing or does not match the information on the tape.
VOL XXXXXX IS SEQUENCE XXX, XXX REQUESTED	An invalid volume sequence is mounted instead of the one that was requested.
WRONG VOLUME: XXXXXX MOUNTED	The wrong volume is mounted. This message also displays the volume ID of the tape that is mounted.
XXXXXX FILE: XXXXXXXX VOL SEQ IS XXX MUST BE 1	An invalid volume sequence number is specified when volume sequence number 1 was expected.

A

access lock The requestor is willing to accept minor inconsistencies of the data while accessing the database (an approximation is sufficient).

An access lock permits modifications on the underlying data while the SELECT operation is in progress.

access module processor (AMP) A virtual processor that receives steps from a parsing engine (PE) and performs database functions to retrieve or update data. Each AMP is associated with one virtual disk, where the data is stored. An AMP manages only its own virtual disk and not the virtual disk of any other AMP.

AMP See [access module processor \(AMP\)](#).

AP Application Processor

APRC application Processor Reset Containment

ARC Teradata's Archive/Recovery Utility

B

BTEQ Basic Teradata Query

C

Call-Level Interface (CLI) A programming interface designed to support SQL access to databases from shrink-wrapped application programs. SQL/CLI provides an international standard implementation-independent CLI to access SQL databases. Client-server tools can easily access database through dynamic link libraries. It supports and encourages a rich set of client-server tools.

conditional expression A user-provided expression needed for a selected partitions operation. A conditional expression specifies which rows or partitions the operation applies to. With PARTITIONS WHERE, the expression is matched against the partitioning expression defined on the specified table. If the range of valid values for a partition overlaps the range of values that match the conditional expression, then the entire partition is selected; otherwise, the entire partition is ignored. The partition does not necessarily need to contain a row that matches the conditional expression in order to be selected. With LOG WHERE, the expression is matched against individual rows; any row that matches the conditional expression (but is not restored to the target table) is inserted into the error table.

CMI Conversational Monitor System

D

database A related set of tables that share a common space allocation and owner. A collection of objects that provide a logical grouping for information. The objects include, tables, views, macros, triggers, and stored procedures.

DBC Database computer

DIP Database DBC Initialization Procedure

E

exclusive lock The requester has exclusive rights to the locked resource. No other process can read from, write to, or access the locked resource in any way.

G

GUI Graphical user interface

I

IFP Interface Processor

I/O Input/output

J

JCL Job Control Language

join A select operation that combines information from two or more tables to produce a result.

K

KB Kilobytes

L

LOB Large object

log A record of events. A file that records events. Many programs produce log files. Often you will look at a log file to determine what is happening when problems occur. Log files have the extension “.log”.

M

MVS Multiple virtual storage

O

Open Database Connectivity (ODBC) Under ODBC, drivers are used to connect applications with databases. The ODBC driver processes ODBC calls from an application, but passes SQL requests to the Teradata Database for processing.

P

partitioning expression The expression defined in the table definition for a table with a partitioned primary index (PPI) that specifies how the data for the table is to be partitioned.

PDE Parallel Database Extension

PE Parsing Engine

PPI Partitioned Primary Index

R

RCC Recovery Control Catalog

read lock Several users can hold read locks on a resource, during which time the system permits no modification of that resource.

Read locks ensure consistency during read operations such as those that occur during a **SELECT** statement.

Relational Database Management System (RDBMS) A database management system in which complex data structures are represented as simple two-dimensional tables consisting of columns and rows. For Teradata SET, RDBMS is referred to as “Teradata Database.”

S

SMP Symmetric Multi-Processing

SQL See [Structured Query Language \(SQL\)](#).

Structured Query Language (SQL) The initials, SQL, are pronounced either *see-kwell* or as separate letters. SQL is a standardized query language for requesting information from a database. SQL consists of a set of facilities for defining, manipulating, and controlling data in a relational database.

T

table A two-dimensional structure made up of one or more columns with zero or more rows that consist of fields of related information. See also database.

TDP Teradata Director Program

U

UDF User-defined function. By providing a mechanism that supports the creation of SQL functions, scalar and aggregate UDFs allow users to add their own extensions to Teradata Database SQL. These functions can be used to operate on any Teradata Database data type and can be utilized wherever built-in functions are used. If a function doesn't exist to perform a specific task or algorithm, you can write one that does. UDFs can also simplify complex SQL syntax.

UDT User-defined type. UDTs introduce object-oriented technology to SQL. Users can define custom data types that model the structures and behaviors of the data in their applications.

V

VM virtual Machine

W

write lock The requester has exclusive rights to the locked resource, except for readers not concerned with data consistency.

Symbols

\$ARC

- database added by LOGSKIPPED 108

- default database 79

* character

- ANALYZE statement 135

A

ABORT keyword

- ARCHIVE statement 139, 144

- BUILD statement 151

- COPY statement 157

- RESTORE statement 190

- ROLLBACK statement 206, 207

- ROLLFORWARD statement 209, 211

access privileges

- ARCHIVE statement 140

- BUILD statement 151

- CHECKPOINT statement 153

- COPY statement 158

- DELETE JOURNAL statement 171

- RELEASE LOCK statement 186

- RESTORE statement 191

- REVALIDATE REFERENCES FOR statement 203

- ROLLBACK statement 206

- ROLLFORWARD statement 209

AccessRights table 25

accid parameter

- LOGON statement 183

AccLogRuleTbl table 25

Accounts table 25

ALL FROM ARCHIVE keywords

- COPY statement 156, 163

- RESTORE statement 189, 200

ALL keyword

- ANALYZE statement 135

- ARCHIVE statement 138

- BUILD statement 150, 151

- CHECKPOINT statement 152

- COPY statement 156

- DELETE DATABASE statement 168

- DELETE JOURNAL statement 170

- RELEASE LOCK statement 185, 186

- RESTORE statement 189, 190

- REVALIDATE REFERENCES FOR statement 202

- ROLLBACK statement 205, 206

- ROLLFORWARD statement 208, 209

ALL PARTITIONS

- potential risks 33

all-amp archives

- selected partitions 146

all-amps restore

- selected partitions 43

AMP keyword

- ARCHIVE statement 139, 142

- COPY statement 163

- RESTORE statement 190, 198

AMPs

- all, restoring 46

- archive configuration 28

- offline

 - archives 36

 - checkpoints 69

- one offline, restoring 48

- recovering specific 54

- recovering while offline 53

- restoring after reconfiguration 51

- restoring offline 48

- restoring specific 52

- restoring with all AMPS online 46

ANALYZE statement

- * character 135

- ALL keyword 135

- CATALOG keyword 135

- character set support 136

- DISPLAY keyword 135

- FILE keyword 135

- kanji support 136

- LONG keyword 135

- VALIDATE keyword 135

APPLY TO keywords

- COPY statement 157, 162

ARC

- return codes 127

- terminology 54

ARC scripts

- samples 22

ARC, Kanji support. See Kanji

ARCDFLT

- about defaults 21

ARCDFLT environment variable 71, 72

ARCENV

- about defaults 21

- ARCENV environment variable 71, 73
 - ARCENVX
 - about defaults 21
 - ARCENVX environment variable 71, 73
 - archive
 - AMPs 28
 - cluster archive 35
 - databases 28
 - dictionary archive
 - general characteristics 28
 - large objects 37
 - locks during archive of selected partitions 30
 - nonhashed tables 37
 - offline AMPs 36
 - online 38
 - potential risks for selected partitions 33
 - procedure for selected partitions 31
 - sample script 22
 - selected partitions of PPI tables 30
 - tables 28
 - ARCHIVE statement
 - ABORT keyword 139, 144
 - access privileges 140
 - ALL keyword 138
 - AMP keyword 139, 142
 - CATALOG operations 80
 - CLUSTER/CLUSTERS keyword 139, 142
 - DATA TABLE/TABLES keywords 138
 - DICTIONARY TABLE/TABLES keywords 138
 - EXCLUDE keyword 138
 - EXCLUDE TABLE/TABLES 138, 157, 190
 - FILE keyword 140
 - GROUP keyword 140, 144
 - INDEXES keyword 139, 144
 - JOURNAL TABLE/TABLES keywords 138
 - NO FALLBACK TABLE/TABLES keywords 138
 - NONEMPTY DATABASE/DATABASES keywords 140, 145
 - referential integrity 142
 - RELEASE LOCK keywords 139
 - USE GROUP READ/USE GROUP READ LOCKS keywords 140
 - ARCMAN
 - defined 17
 - required files for 17
 - starting
 - from MVS 18
 - from VM 19, 85
 - from Windows 20
 - starting Teradata ARC 16
 - ARCMAN statements
 - ARCHIVE 137
 - BUILD 150
 - CHECKPOINT 152
 - COPY 155
 - DELETE DATABASE 168
 - DELETE JOURNAL 170
 - LOGOFF 182
 - LOGON 183
 - RELEASE LOCK 185
 - RESTORE 188
 - REVALIDATE REFERENCES FOR 202
 - ROLLBACK 204, 205
 - ROLLFORWARD 208
- ## B
- BACKUP NOT DOWN keywords
 - RELEASE LOCK statement 186, 187
 - batch mode 16
 - BUILD DATA TABLES operation 49
 - BUILD statement
 - ABORT keyword 151
 - access privileges 151
 - ALL keyword 150, 151
 - DATA TABLE/TABLES keywords 150
 - EXCLUDE keyword 150
 - JOURNAL TABLES keywords 150
 - NO FALLBACK TABLE/TABLES keywords 150
 - RELEASE LOCK keywords 151
- ## C
- cancel operation 22
 - CATALOG
 - creating multiple tables 80
 - default database 79
 - primary index in 80
 - CATALOG keyword
 - ANALYZE statement 135
 - CATALOG runtime parameter
 - defined 75
 - syntax 78
 - change images
 - restored 48
 - character set support
 - ANALYZE statement 136
 - COPY statement 166
 - character set, establishing 84
 - CHARSETNAME runtime parameter
 - defined 75
 - syntax 83
 - CHECKPOINT runtime parameter
 - defined 75
 - syntax 88
 - CHECKPOINT statement
 - access privileges 153
 - ALL keyword 152
 - chkptname parameter 153

- EXCLUDE keyword 152
- NAMED keyword 153, 154
- USE ACCESS LOCK keywords 153
- USE LOCK keywords 153
- USE READ LOCK keywords 153
- WITH SAVE keywords 153
- CHECKSUM runtime parameter
 - defined 75
 - syntax 90
- chkptname parameter
 - CHECKPOINT statement 153
 - ROLLBACK statement 206, 207
 - ROLLFORWARD statement 209, 210
- cluster archive
 - NO BUILD option 51
 - restoring 49
- cluster archive. See archive.
- CLUSTER/CLUSTERS keyword
 - ARCHIVE statement 139, 142
 - COPY statement 163
 - RELEASE LOCK statement 186
 - RESTORE statement 190, 199
- CMS tape support messages 229
- CollationTbl table 25
- command-line
 - saving as config file or variable 20
- concurrency control 29
- conditional expression
 - considerations during archive of selected partitions 31
 - example 31
- config file
 - creating from command-line options 20
- copy
 - database 58
 - large objects 56
 - restoring undefined tables 40
- COPY statement
 - ABORT keyword 157
 - access privileges 158
 - ALL FROM ARCHIVE keywords 156, 163
 - ALL keyword 156
 - AMP keyword 163
 - APPLY TO keywords 157, 162
 - character set support 166
 - CLUSTER/CLUSTERS keyword 163
 - DATA TABLE/TABLES keywords 156
 - DBC.DBCAssociation table 160
 - DICTIONARY TABLE/TABLES keywords 156
 - examples 56, 159
 - EXCLUDE keyword 156
 - FILE keyword 157
 - FROM keyword 157, 161
 - HUT locks 160
 - JOURNAL TABLE/TABLES keywords 156

- kanji support 166
- macros 160
- NO BUILD keywords 157, 163
- NO FALLBACK keywords 157, 161
- NO FALLBACK TABLE/TABLES keywords 156
- NO JOURNAL keywords 157
- referential integrity 161
- RELEASE LOCK keywords 157
- REPLACE CREATOR keywords 157
- stored procedures 160
- triggers 160
- views 160
- WITH JOURNAL TABLE keywords 157, 162
- CREATE INDEX keyword 40
- creating tables for ARCHIVE statements 80
- CURRENT keyword
 - ROLLFORWARD statement 209

D

- data dictionary definitions
 - restoration of 39
- data extension modules 23, 172, 174
- DATA TABLE/TABLES keywords
 - ARCHIVE statement 138
 - BUILD statement 150
 - COPY statement 156
 - RESTORE statement 189
- database archive. See archive.
- database DBC
 - failed restores 41
 - restoring 41
- database SYSUDTLIB 27
- databases
 - copying 54
- DATAENCRYPTION runtime parameter
 - defined 75
 - syntax 91
- DBase table 25
- DBC database. See database DBC.
- DBC Initialization Procedure (DIP)
 - during restore with one AMP offline 48
- DBC.DBCAssociation table
 - COPY statement 160
- DBCLOG
 - starting from MVS 18
- DBCPFX
 - starting from MVS 18
- DBSERROR command line parameter 129
- DEFAULT runtime parameter
 - defined 75
 - syntax 93
- definition of Teradata ARC 3, 15
- DELETE DATABASE statement

- ALL keyword 168
- EXCLUDE keyword 168
- DELETE JOURNAL statement
 - ALL keyword 170
 - EXCLUDE keyword 170
 - RESTORED keyword 170
 - SAVED keyword 170
- DELETE keyword
 - ROLLBACK statement 206, 207
 - ROLLFORWARD statement 209, 211
- DELETE statement 46
- DEMODULE parameter 23, 95
- DEPARM parameter 23, 97
- dictionary archive. See archive.
- DICTIONARY TABLE/TABLES keywords
 - ARCHIVE statement 138
 - COPY statement 156
 - RESTORE statement 189, 198
- DISPLAY keyword
 - ANALYZE statement 135
- DROP keyword 40
- dropped tables
 - restoring 41
- DUMP
 - keyword to start on MVS 18
- DUMP, See ARCHIVE.

E

- ENABLE DATA EXTENSION statement 172, 174
- ENABLE ENCRYPTION statement 174
- encryption module 174
- environment variables
 - ARCDFLT 71
 - ARCENV 71
 - ARCENVX 71
 - creating from command-line options 20
 - override priority 21
- ERRLOG runtime parameter
 - defined 76
 - syntax 99
- error tables
 - characteristics 43
- ERRORDB keyword
 - REVALIDATE REFERENCES FOR statement 203
- eventno parameter
 - ROLLFORWARD statement 209
- EXCLUDE keyword
 - ARCHIVE statement 138
 - BUILD statement 150
 - CHECKPOINT statement 152
 - COPY statement 156
 - DELETE DATABASE statement 168
 - DELETE JOURNAL statement 170

- RELEASE LOCK statement 185
- RESTORE statement 190
- REVALIDATE REFERENCES FOR statement 202
- ROLLBACK statement 206
- ROLLFORWARD statement 208
- EXCLUDE TABLE/TABLES keywords
 - ARCHIVE statement 138, 157, 190
- EXCLUDE TABLES 43

F

- failed restores 41
- fallbacks
 - cluster archive 49
 - restoring 48
- FATAL runtime parameter
 - defined 76
 - syntax 100
- FILE keyword
 - ANALYZE statement 135
 - ARCHIVE statement 140
 - COPY statement 157
- FILEDEF runtime parameter
 - defined 76
 - syntax 101
- FROM keyword
 - COPY statement 157, 161
- full-table locks
 - during archive of selected partitions 30

G

- GROUP keyword
 - ARCHIVE statement 140, 144
- group read locks
 - restoring selected partitions 62

H

- HALT runtime parameter
 - defined 76
 - syntax 103
- hash functions
 - of archive/restore 40
- HELP DATABASE keyword 40
- HEX runtime parameter
 - defined 76
 - syntax 104
- host utility locks. See HUT locks
- Hosts table 26
- HUT locks
 - archiving and 61
 - COPY statement 160
 - other operations and 64
 - restoring 63

using 60

I

incremental archives 30

indexes

limitation on archive/restore 40

secondary tables 40

unique secondary 48

INDEXES keyword

ARCHIVE statement 139, 144

insufficient memory

during restore 40

for large tables 40

interactive mode 16

interrupted restores 41

IOMODULE runtime parameter

defined 76

syntax 105

IOPARM runtime parameter

defined 76

syntax 106

J

join indexes

limitation on archive/restore 40

JOURNAL TABLE/TABLES keywords

BUILD statement 150

COPY statement 156

RESTORE statement 189

journal tables

archiving 66, 67

change data location 65

checkpoint operations, controlling 68

creating 65

data location 65

local 67

remote 67

setting up 64

subtable

active 66

restored 66

saved 66

K

kanji

ARC support for 85

kanji support

ANALYZE statement 136

COPY statement 166

keywords

ABORT 144, 207

ALL FROM ARCHIVE 200

AMP 142, 163, 198

APPLY TO 162

BACKUP NOT DOWN 187

CLUSTER 142, 163, 199

DELETE 207, 211

DICTIONARY TABLE 198

FROM 161

GROUP 144

INDEXES 144

NAMED 154

NO BUILD 163, 200

NO FALLBACK 161

NONEMPTY DATABASE 145

PRIMARY DATA 210

RELEASE LOCK 210

RESTORE FALLBACK 199

USE LOCK 153

WITH JOURNAL TABLE 162

WITH SAVE 153

L

large objects (LOBs)

archiving 37

copying 56

restoring 40

limitations

character set 85

Linux RedHat/SuSE support 15

local journaling. See journal tables

locks

during an all-AMP restore 48

in cluster archives 49

in copy operations 56

restoring 63

locks. See HUT locks

locks. See utility locks.

LOG WHERE 43

LOGGING ONLINE ARCHIVE OFF statement 177

LOGGING ONLINE ARCHIVE ON statement 179

logical space 51

LOGOFF statement 182

LOGON runtime parameter

defined 76

syntax 107

LOGON statement

accid parameter 183

password parameter 183

tdpid parameter 183

userid parameter 183

LogonRuleTbl table 26

LOGSKIPPED runtime parameter

defined 76

syntax 108

LONG keyword
 ANALYZE statement 135

M

macros
 COPY statement 160
 migration scripts 41
 MODIFY DATABASE statement 41
 MODIFY USER statement 41
 multiple CATATLOG tables 80
 MVS
 starting ARCMAN 18

N

NAMED keyword
 CHECKPOINT statement 153, 154
 NetBackup
 using with Teradata ARC 20
 Next table 26
 NO BUILD keywords
 COPY statement 157, 163
 RESTORE statement 190, 200
 NO BUILD option 51
 NO DELETE keywords
 ROLLBACK statement 206
 ROLLFORWARD statement 209
 NO FALLBACK keywords
 COPY statement 157, 161
 NO FALLBACK TABLE/TABLES keywords
 ARCHIVE statement 138
 BUILD statement 150
 COPY statement 156
 RESTORE statement 189
 NO JOURNAL keywords
 COPY statement 157
 NONEMPTY DATABASE/DATABASES keywords
 ARCHIVE statement 140, 145

O

offline AMPs 36
 OLDCATALOG parameter 80
 OldPasswords table 26
 online archiving 38
 OUTLOG runtime parameter
 defined 76
 syntax 109
 OVERRIDE keyword
 RELEASE LOCK statement 186
 overrides
 by ARCENV 21
 overview 3, 15
 Owners table 26

P

Parents table 26
 PARM runtime parameter
 defined 76
 syntax 110
 partitioning. See selected partitions.
 PARTITIONS BY 43
 PARTITIONS WHERE keyword
 define rows for archive 146
 potential risks 33
 restore partitioned data 43
 password parameter
 LOGON statement 183
 PAUSE runtime parameter
 defined 77
 syntax 113
 PERFFILE runtime parameter
 defined 77
 syntax 114
 perm space
 affecting restore 51
 physical space 51
 platforms 15
 PPI tables
 archiving selected partitions 30
 PRIMARY DATA keywords
 ROLLBACK statement 207
 ROLLFORWARD statement 206, 209, 210
 primary index
 in CATALOG table 80
 product version numbers 3
 Profiles table 26

Q

query band 212

R

RCC
 RCConfiguration table 26, 221
 RCEvent table 26, 219
 RCMedia table 26, 222
 RCConfiguration table 26, 221
 RCEvent table 26, 219
 RCMedia table 26, 222
 reconfiguration
 restoring after 51
 recovering
 general characteristics 52
 offline AMPs 53
 specific AMP 54
 tables and databases 52
 referential integrity

- after all-AMPs copy 161
- ARCHIVE statement 142
- RESTORE statement 194
- reinitialization prior to restore 41
- RELEASE LOCK keywords
 - ARCHIVE statement 139
 - BUILD statement 151
 - COPY statement 157
 - RESTORE statement 190
 - REVALIDATE REFERENCES FOR statement 202
 - ROLLBACK statement 206
 - ROLLFORWARD statement 209, 210
- RELEASE LOCK statement
 - access privileges 186
 - ALL keyword 185, 186
 - BACKUP NOT DOWN keywords 186, 187
 - CLUSTER/CLUSTERS keyword 186
 - EXCLUDE keyword 185
 - OVERRIDE keyword 186
- remote journaling. *See* journal tables
- RENAME keyword 40
- RepGroup table 26
- REPLACE CREATOR keywords
 - COPY statement 157
- RESTART runtime parameter
 - defined 77
 - syntax 116
- RESTARTLOG runtime parameter
 - defined 77
 - syntax 117
- restore
 - after AMP reconfiguration 51
 - all databases, example 47
 - change images 48
 - cluster archive 49
 - conditions of 39
 - considerations before 40
 - data dictionaries 39
 - database DBC 41
 - defined 39
 - difference in perm space 51
 - DIP procedure
 - dropped items 40
 - dropped tables 41
 - dropping objects before 41
 - entire database 40, 42
 - error tables for selected partitions 43
 - example for selected partitions 32
 - EXCLUDE option 47
 - failed or interrupted 41
 - fallbacks 48
 - full database from selected table archive 42
 - general characteristics 39
 - large objects 40
 - locks 39
 - migration script after 41
 - NO BUILD option 51
 - permanent journal tables 41
 - potential risks for selected partitions 33
 - sample script 23
 - selected partitions 42
 - selected table 42
 - specific AMP 52
 - types of 42
- restore database DBC 27
- restore dropped 41
- RESTORE FALLBACK keywords
 - RESTORE statement 190, 199
- RESTORE statement
 - ABORT keyword 190
 - access privileges 191
 - ALL FROM ARCHIVE keywords 189, 200
 - ALL keyword 189, 190
 - AMP keyword 190, 198
 - CLUSTER/CLUSTERS keyword 190, 199
 - DATA TABLE/TABLES keywords 189
 - DICTIONARY TABLE/TABLES keywords 189, 198
 - EXCLUDE keyword 190
 - JOURNAL TABLE/TABLES keywords 189
 - NO BUILD keywords 190, 200
 - NO FALLBACK TABLE/TABLES keywords 189
 - referential integrity 194
 - RELEASE LOCK keywords 190
 - RESTORE FALLBACK keywords 190, 199
- RESTORED keyword
 - DELETE JOURNAL statement 170
 - ROLLFORWARD statement 209
- resume operation. *See* cancel operation.aborted operation.
 See cancel operation.
- REVALIDATE REFERENCES FOR statement
 - access privileges 203
 - ALL keyword 202
 - ERRORDB keyword 203
 - example 204
 - EXCLUDE keyword 202
 - RELEASE LOCK keywords 202
- RoleGrants table 26
- Roles table 26
- ROLLBACK statement
 - ABORT keyword 206, 207
 - access privileges 206
 - ALL keyword 205, 206
 - chkptname parameter 206, 207
 - DELETE keyword 206, 207
 - EXCLUDE keyword 206
 - NO DELETE keywords 206
 - RELEASE LOCK keywords 206
 - USE CURRENT JOURNAL keywords 206

- USE RESTORED JOURNAL keywords 206
- rollforward
 - cluster archive 49
- ROLLFORWARD statement
 - ABORT keyword 209, 211
 - access privileges 209
 - ALL keyword 208, 209
 - chkptname parameter 209, 210
 - CURRENT keyword 209
 - DELETE keyword 209, 211
 - eventno parameter 209
 - EXCLUDE keyword 208
 - NO DELETE keywords 209
 - PRIMARY DATA keywords 206, 209, 210
 - RELEASE LOCK keywords 209, 210
 - RESTORED keyword 209
- runtime parameters
 - CATALOG 78
 - CHARSETNAME 83
 - CHECKPOINT 88
 - CHECKSUM 90
 - DATAENCRYPTION 91
 - DEFAULT 93
 - DEMODULE 95
 - DEPARM 97
 - ERRLOG 99
 - FATAL 100
 - FILEDEF 101
 - HALT 103
 - HEX 104
 - IOMODULE 105
 - IOPARM 106
 - LOGON 107
 - LOGSKIPPED 108
 - on MVS systems 75
 - on VM systems 75
 - OUTLOG 109
 - PARM 110
 - PAUSE 113
 - PERFFILE 114
 - RESTART 116
 - RESTARTLOG 117
 - SESSIONS 118
 - specifying order of 75
 - STARTAMP 120
 - UEN 101, 121
 - VERBOSE 122
 - WORKDIR 126

S

- sample scripts
 - archive 22
 - restore 23

- SAVED keyword
 - DELETE JOURNAL statement 170
- scripts
 - samples 22
- secondary indexes
 - during a restore without fallback 48
- secondary table indexes 40
- selected partition archive
 - procedures 31
- selected partitions
 - all-amps restore 43
 - archive 30
 - archiving PPI tables 146
 - error tables 43
 - full-table locks 30
 - group read locks 62
 - keywords for 146
 - limitations 146
 - locks 39
 - PARTITIONS WHERE keyword 146
 - potential risks 33
 - restoring 42
 - UtilVersion match 40
- selected table
 - restoring 42
- session query band 212
- SESSIONS runtime parameter
 - defined 77
 - syntax 118
- SET QUERY_BAND statement 212
- SHOW JOIN INDEX keyword 40
- SIGILL (illegal signal) 132
- SIGINT (interrupt signal) 132
- SIGSEGV (segmentation violation signal) 132
- SIGTERM (terminate signal) 132
- Single Sign-On (SSO) 183, 184
- SkippedTables table 108
- software releases
 - supported 3
- specifications 16
- SQL CREATE keyword 40
- SSO (Single Sign-On) 183, 184
- STARTAMP runtime parameter
 - defined 77
 - syntax 120
- starting Teradata ARC 16
- statistics
 - during archive of selected partitions 31
- stored procedures
 - COPY statement 160
- supported platforms 15
- syntax, how to read 223
- SysSecDefaults table 26
- system tables 25

system upgrades
 before restore 42

T

table indexes 40
 table-level exclude
 ARCHIVE statement 147
 tables
 copying 54
 dropped during restore 40
 insufficient memory 40
 nonhashed, archiving 37
 permanent journal tables 41
 PPI tables 30
 recovering 52
 restore of undefined 40
 restoring tables with fallbacks 48
 restoring tables without fallbacks 48
 statistics during archive of selected partitions 31
 tdpid parameter
 LOGON statement 183
 Teradata Relational Database Management System. See
 Teradata Database
 terms used by Teradata ARC 16
 Translation table 26
 triggers
 COPY statement 160
 troubleshooting
 risks for selected partitions 33

U

UDTCast table 26
 UDTInfo table 26
 UDTTransform table 26
 UEN runtime parameter
 defined 77
 in a file name 101
 syntax 121
 undefined tables
 restored with COPY 40
 UNIX signals
 SIGILL 132
 SIGINT 132
 SIGSEGV 132
 SIGTERM 132
 upgrade
 need for conversion utilities 42
 of systems 42
 USE ACCESS LOCK keywords
 CHECKPOINT statement 153
 USE CURRENT JOURNAL keywords
 ROLLBACK statement 206
 USE GROUP READ keywords

 ARCHIVE statement 140
 USE GROUP READ LOCKS keywords
 ARCHIVE statement 140
 USE LOCK keywords
 CHECKPOINT statement 153
 USE READ LOCK keywords
 CHECKPOINT statement 153
 USE RESTORED JOURNAL keywords
 ROLLBACK statement 206
 userid parameter
 LOGON statement 183
 users
 dropped during restore 40
 Utility Event Number. See runtime parameters, UEN
 utility locks
 during restores 39
 UtilVersion
 matching for selected partitions 40

V

VALIDATE keyword
 ANALYZE statement 135
 variables
 creating from command-line options 20
 override priority 21
 VERBOSE runtime parameter
 defined 77
 syntax 122
 verifying version number 15
 Veritas NetBackup
 using with Teradata ARC 20
 version numbers 3, 15
 views
 COPY statement 160
 dropped during restore 40
 VM
 starting ARCMAN 19

W

Windows
 starting ARCMAN 20
 WITH JOURNAL TABLE keywords
 COPY statement 157, 162
 WITH SAVE keywords
 CHECKPOINT statement 153
 WORKDIR runtime parameter
 defined 77
 syntax 126
 write locks
 during a restore of selected partitions 39

