

UniVerse

User Reference

Version 10.3
February, 2009

IBM Corporation
555 Bailey Avenue
San Jose, CA 95141

Licensed Materials – Property of IBM

© Copyright International Business Machines Corporation 2008, 2009. All rights reserved.

AIX, DB2, DB2 Universal Database, Distributed Relational Database Architecture, NUMA-Q, OS/2, OS/390, and OS/400, IBM Informix®, C-ISAM®, Foundation.2000™, IBM Informix® 4GL, IBM Informix® DataBlade® module, Client SDK™, Cloudscape™, Cloudsync™, IBM Informix® Connect, IBM Informix® Driver for JDBC, Dynamic Connect™, IBM Informix® Dynamic Scalable Architecture™ (DSA), IBM Informix® Dynamic Server™, IBM Informix® Enterprise Gateway Manager (Enterprise Gateway Manager), IBM Informix® Extended Parallel Server™, i.Financial Services™, J/Foundation™, MaxConnect™, Object Translator™, Red Brick® Decision Server™, IBM Informix® SE, IBM Informix® SQL, InformiXML™, RedBack®, SystemBuilder™, U2™, UniData®, UniVerse®, wIntegrate® are trademarks or registered trademarks of International Business Machines Corporation.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Windows, Windows NT, and Excel are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names used in this publication may be trademarks or service marks of others.

This product includes cryptographic software written by Eric Young (eay@cryptosoft.com).

This product includes software written by Tim Hudson (tjh@cryptosoft.com).

Documentation Writer: Claire Gustafson, Shelley Thompson, Anne Waite

US GOVERNMENT USERS RESTRICTED RIGHTS

Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Table of Contents

Preface

Organization of This Manual	xxii
UniVerse Documentation.	xxv
Related Documentation	xxviii
Client API Documentation	xxix

Chapter 1

UniVerse Commands

.A	1-11
.C	1-13
.D	1-16
.I	1-18
.L	1-19
.R	1-21
.S	1-23
.U	1-25
.X	1-26
.?	1-28
*	1-29
<<...>>	1-31
ABORT	1-34
ABORT.LOGIN	1-35
ACCOUNT.FILE.STATS	1-36
ACTIVATE.ENCRIPTION.KEY	1-38
ACTLIST	1-39
ANALYZE.FILE	1-41
analyze.shm	1-45
ANALYZE.SHM	1-55
ASSIGN	1-60
AUTOLOGOUT	1-63
AVAIL	1-64
BASIC	1-66
BELL	1-69

BLOCK.PRINT	1-70
BLOCK.TERM	1-72
BREAK	1-74
BUILD.INDEX	1-75
CATALOG	1-77
CD	1-83
CENTURY.PIVOT	1-85
CHAP	1-87
CHANGE.DOMAIN.	1-88
CHDIR	1-90
CHECK.SUM	1-91
CLEAN.ACCOUNT.	1-94
CLEAR.FILE	1-97
CLEAR.LOCKS	1-99
CLEARCOMMON	1-101
CLEARDATA	1-102
CLEARPROMPTS	1-103
CLEARSELECT	1-104
CLR	1-105
CNAME	1-106
COMO	1-109
COMPILE.DICT	1-111
COMPILE.DICTS	1-113
CONFIG	1-114
CONFIGURE.FILE	1-117
CONNECT.	1-122
CONVERT.ACCOUNT.	1-128
CONVERT.SQL	1-129
CONVERT.VOC	1-140
COPY	1-142
COPY.LIST	1-147
CORE	1-149
COUNT.	1-150
CP	1-152
CREATE.ENCRYPTION.KEY	1-154
CREATE.FILE	1-156
CREATE.INDEX.	1-165
CREATE.LDIR	1-169
CREATE.LFILE	1-171
CS	1-173
CSH	1-174
CT	1-176

DATA	1-178
DATE	1-180
DATE.FORMAT	1-181
DEACTIVATE.ENCRYPTION.KEY	1-184
DEACTLIST	1-185
DECATALOG	1-186
DECRYPT.FILE	1-187
DEFINE.DF	1-191
DEL.RFILE	1-197
DELETE	1-199
DELETE.CATALOG	1-201
DELETE.ENCRYPTION.KEY	1-203
DELETE.FILE	1-205
DELETE.INDEX	1-208
DELETE.LFILE	1-209
DELETE.LIST	1-211
DISABLE.DECRYPTION	1-212
DISABLE.INDEX	1-213
DISPLAY.	1-215
DIVERT.OUT	1-216
DLIST.	1-219
DOS	1-221
ED	1-222
EDIT.CONFIG	1-223
EDIT.LIST	1-226
ENABLE.DECRYPTION	1-227
ENABLE.INDEX	1-228
ENABLE.RECOVERY	1-229
ENCRYPT.FILE	1-231
ENVIRONMENT or ENV	1-236
ESEARCH	1-238
EXCHANGE	1-239
FANCY.FORMAT	1-240
FILE.STAT	1-241
FILE.USAGE	1-243
FILE.USAGE.CLEAR	1-246
FILE.USAGE.OFF.	1-247
FORM.LIST	1-248
FORMAT.	1-250
FORMAT.CONV	1-252
GET.FILE.MAP	1-258
GET.LIST	1-260

GET.LOCALE	1-262
GET.STACK	1-264
GET.TERM.TYPE	1-265
GO	1-268
GRANT.ENCRYPTION.KEY	1-270
GROUP.STAT	1-272
GROUP.STAT.DETAIL	1-274
HASH.AID	1-276
HASH.HELP	1-280
HASH.HELP.DETAIL	1-282
HASH.TEST	1-284
HASH.TEST.DETAIL	1-286
HELP	1-289
HUSH	1-292
IAM	1-293
IF	1-295
INITIALIZE.CATALOG	1-297
INITIALIZE.DEMO	1-299
JOBS	1-300
LIMIT	1-301
LIST	1-302
LIST.DF	1-306
LIST.DIFF	1-307
LIST.ENCRYPTION.FILE	1-310
LIST.ENCRYPTION.KEY	1-311
LIST.ENCRYPTION.WALLET	1-312
LIST.FILE.STATS	1-313
LIST.INDEX	1-315
LIST.INTER	1-319
LIST.ITEM	1-322
LIST.LABEL	1-325
LIST.LOCALES	1-330
LIST.LOCKS	1-332
LIST.MAPS	1-334
LIST.READU	1-336
LIST.SICA	1-341
LIST.UNION	1-344
LISTME	1-347
LISTU	1-349
LIST	1-351
LOCK	1-353
LOG.RESTORE	1-355

LOG.SAVE	1-357
LOGIN	1-359
LOGON	1-361
LOGOUT.	1-362
LOGTO	1-364
LOGTO.ABORT	1-365
LOOP	1-366
MAIL	1-368
MAKE.DEMO.FILES	1-371
MAKE.DEMO.TABLES	1-372
MAKE.MAP.FILE	1-373
MAP	1-375
MASTER.	1-376
MENU.DOC.	1-377
MENU.PIX	1-378
MENUS	1-379
MERGE.LIST	1-380
MESSAGE	1-383
MKFILELIST	1-385
MOTIF	1-387
NLS.ADMIN	1-390
NLS.UPDATE.ACCOUNT	1-391
NOTIFY	1-393
NSELECT	1-394
OFF	1-396
ON.ABORT	1-398
ON.EXIT.	1-399
P.ATT	1-400
P.DET	1-401
PAGE.MESSAGE	1-402
PASSWD.	1-403
PHANTOM	1-404
PORT.STATUS	1-407
PRIME	1-411
PRINT.ADMIN	1-412
PTERM (UNIX)	1-413
PTERM (Windows Platforms)	1-430
PTIME	1-438
QSELECT	1-439
QUIT	1-441
RADIX	1-442
RAID	1-446

RAID.GUI	1-449
REBUILD.DF	1-452
RECORD	1-456
RECOVERY.CHECKPOINT	1-460
RECOVERY.CONSISTENT	1-462
REFORMAT	1-463
RELEASE	1-468
RELEASE.LFILE	1-469
REMOVE.DEMO.FILES	1-471
REMOVE.DEMO.TABLES	1-472
REPEAT	1-473
RESIZE	1-474
RESTORE.LOCALE	1-479
REVISE	1-480
REVOKE.ENCRIPTION.KEY	1-482
RUN	1-483
SAVE.LIST	1-485
SAVE.LOCALE	1-487
SAVE.STACK	1-488
SEARCH	1-489
SELECT	1-492
SEMAPHORE.STATUS	1-495
SET.FILE	1-497
SET.FILE.MAP	1-498
SET.GCI.MAP	1-501
SET.INDEX	1-503
SET.LOCALE	1-507
SET.LOG.ATTR	1-509
SET.REMOTE.ID	1-511
SET.SEQ.MAP	1-513
SET.SQL	1-515
SET.TERM.TYPE	1-518
SETFILE	1-521
SETPTR (UNIX)	1-522
SETPTR (Windows Platforms)	1-530
SETPTR.DEFAULT	1-539
SETUP.DEMO.SCHEMA	1-540
SH	1-541
SHUTDOWN.RECOVERY	1-543
SLEEP	1-545
SORT	1-547
SORT.ITEM	1-551

SORT.LABEL	1-554
SP.ASSIGN (UNIX)	1-559
SP.ASSIGN (Windows Platforms)	1-561
SP.EDIT (UNIX)	1-562
SP.EDIT (Windows Platforms)	1-567
SP.TAPE	1-571
SPOOL	1-573
SPOOL (Windows Platforms)	1-576
SREFORMAT	1-578
SSELECT	1-583
STAT	1-586
STATUS	1-588
SUM	1-589
SUSPEND.FILES	1-591
SUSPEND.RECOVERY	1-593
T.ATT	1-594
T.BCK.	1-596
T.DET.	1-597
T.DUMP	1-598
T.EOD.	1-601
T.FWD	1-602
T.LOAD	1-603
T.RDLBL.	1-605
T.READ	1-607
T.REW	1-609
T.SPACE	1-610
T.UNLOAD	1-611
T.WEOF	1-612
T.WTLBL	1-613
TANDEM	1-616
TERM.	1-619
TIME	1-622
UMASK	1-623
UNASSIGN	1-626
UNICODE.FILE	1-627
UNLOCK	1-629
UNLOCK SEMAPHORE	1-631
UPDATE.ACCOUNT	1-633
UPDATE.INDEX	1-635
usa	1-637
usd	1-640
USERS	1-642

usm	1-643
usp	1-645
uv.	1-647
UV.LOGIN.	1-649
UV.VI	1-651
uvbackup (UNIX).	1-653
uvbackup (Windows Platforms)	1-657
uvdlockd	1-660
uvfixfile.	1-662
UVFIXFILE	1-669
UVHELP	1-677
uvlictool.	1-679
UVPROMPT	1-680
uvrestore (UNIX).	1-681
uvrestore (Windows Platforms)	1-685
VCATALOG	1-689
VERIFY.DF	1-691
VERIFY.SQL	1-693
VI.	1-699
VLIST	1-701
VVOC	1-703
WHO	1-707

Chapter 2 **UniVerse Keywords**

Introduction	2-12
Keyword Symbols	2-12
Field Modifier Keywords.	2-13
Report Qualifier Keywords	2-15
Keyword Descriptions	2-17
(...)	2-17
@ASSOC_ROW	2-17
1NF	2-17
32BIT.	2-17
64BIT.	2-18
A	2-18
–ACCEPT	2-18
ADDING.	2-18
AFTER	2-18
ALL	2-18
ALL.MATCH	2-19
AND	2-19

ANY	2-19
APPEND	2-20
ARCHIVE	2-20
ARE	2-20
AS	2-20
–AS	2-20
ASSOC	2-21
ASSOC.WITH	2-21
AT	2-22
–AT	2-22
AUTONL	2-22
AUX.PORT	2-22
AUXMAP	2-22
AVERAGE	2-22
AVG	2-23
BANNER	2-23
BASE	2-24
BASIC	2-24
BCI	2-24
BEFORE	2-24
BLK	2-24
BLOCK	2-24
BREAK	2-25
BREAK.ON	2-25
BREAK.SUP	2-27
BRIEF	2-27
BY	2-28
BY-DSND	2-28
BY.DSND	2-28
BY-EXP	2-29
BY.EXP	2-29
BY-EXP-DSND	2-31
BY.EXP.DSND	2-31
CALC	2-33
CALCULATE	2-33
CANCEL	2-33
–CANCEL	2-34
CATALOG	2-34
CHECKPOINT	2-34

CLEAR	2-34
COL.HDG	2-34
COL-HDR-SUPP	2-35
COL.HDR.SUPP	2-35
COL.SPACES	2-35
COL.SPCS	2-35
COL-SUPP	2-35
COL.SUP	2-36
COLLATE	2-36
COMPLETE.	2-36
CONCURRENT	2-36
CONV	2-36
COPIES	2-37
-COPIES.	2-37
COUNT.SUP	2-37
CRT	2-37
CTYPE	2-38
DATA	2-38
DBL.SPC.	2-38
DEFAULT	2-38
DEFAULTS	2-39
DEFER	2-39
DELETE	2-39
DELETING	2-39
DET-SUPP	2-39
DET.SUP.	2-40
DETAIL	2-40
DEVICE	2-40
DEVICELIST	2-40
DICT	2-40
DIFF	2-41
DISABLE LOCK.HIST	2-41
DISKS	2-41
DISPLAY	2-41
DISPLAY.LIKE.	2-41
DISPLAY.NAME	2-42
DOWN	2-42
DYNAMIC	2-42
EJECT	2-42

EMPTY.NULL	2-42
ENABLE LOCK.HIST	2-42
ENDPAGE.	2-43
ENUMERATE	2-43
EQ	2-44
EQUAL	2-44
EVAL	2-44
EVERY.	2-45
EXPLODE.	2-45
EXTERNAL	2-45
FILE.	2-46
FILE.OFF	2-46
FILE.ON	2-46
FILELOCK	2-46
FILEMAP	2-46
FIRST	2-46
FIX	2-47
FMT	2-47
FOOTING	2-48
FOR.	2-50
FORCE.	2-50
FORM	2-50
–FORM.	2-50
FORM.FEED	2-51
–FORM.FEED	2-51
FROM	2-51
FTN	2-51
FUNDAMENTAL	2-52
GE	2-52
GEN.	2-52
GENERAL	2-52
GRAND-TOTAL.	2-52
GRAND.TOTAL.	2-52
GREATER.	2-53
GROUP.SIZE	2-53
GROUPLOCK	2-54
GT	2-54
HDR-SUPP	2-54
HDR.SUP	2-54

-HEAD	2-54
HEADER	2-54
HEADING	2-55
-HEX	2-57
HOLD.	2-57
HUSH.	2-57
ID.ONLY.	2-58
ID-SUP	2-58
ID-SUPP	2-58
ID.SUP	2-58
IN	2-58
IN2	2-58
IN2.FORMAT	2-59
INFO	2-59
INFORM.	2-59
INFORMATION	2-59
INFORMATION.FORMAT	2-59
INODE	2-59
INPLACE	2-60
INQUIRING.	2-60
INTERNAL	2-60
INTERSECT	2-60
INVERT	2-60
INVISIBLE	2-60
IS.NULL	2-61
IS.NOT.NULL	2-61
ISOLATION.	2-61
JOIN.BUFFER	2-61
KEEP	2-62
KEEP.COMMON	2-62
-L	2-62
LARGE.RECORD.	2-62
LE	2-63
LENGTH.	2-63
LESS	2-63
LIKE	2-63
LIST	2-63
-LIST.	2-63
LNUM	2-63

LOCAL	2-64
LOCK.HIST	2-64
LOCK.WAIT	2-64
LOCKS	2-64
LPTR	2-64
LT	2-65
MAP	2-65
MARGIN	2-65
MATCHES	2-65
MATCHING	2-65
MAX	2-68
MERGE.LOAD	2-69
MFILE.HIST	2-69
MIN	2-69
MINIMIZE.SPACE	2-70
MINIMUM.MODULUS	2-71
–MODIFY	2-71
MONETARY	2-71
MTU	2-71
MULTI.VALUE	2-72
MVDISPLAY	2-73
NE	2-73
NEEDNL	2-73
NETWORK	2-73
NEW.PAGE	2-73
NEWACC	2-73
NEXT	2-73
NEXT.AVAILABLE.	2-74
NFMT	2-74
NHEAD	2-74
NO.INDEX	2-75
NO.MATCH	2-75
NO.NEW	2-75
NO.NULLS	2-75
NO.PAGE	2-75
NO.SELECT	2-75
NO.SPLIT	2-76
NO.WAIT	2-76
NO.WARN	2-76

NODE.	2-76
NODEFAULT	2-76
NOEJECT	2-77
NOFMT	2-77
NOHEAD	2-77
-NOHEAD	2-77
NOHOLD	2-77
NOKEEP.	2-77
NONE.	2-77
NOPAGE.	2-78
NOT	2-78
NOT.MATCHING	2-78
NOXREF.	2-80
NULL.	2-80
NUM.SUP	2-81
ODBC.CONNECTIONS	2-81
OF	2-81
OFF	2-81
ON.	2-81
ONLY.	2-82
OPTIM.SCAN	2-82
OVERWRITING	2-82
PCT	2-82
PDICT	2-83
PERCENT	2-83
PERCENT.GROWTH.	2-83
PERCENTAGE.	2-83
PICK	2-84
PICK.FORMAT	2-84
PID	2-84
PIOPEN	2-84
PORT	2-85
-PORTNO	2-85
PREFIX	2-85
PRINT	2-85
PRINTER	2-85
PROGRAMSIZE	2-86
PROMPT.	2-87
-RANGE.	2-87

READLLOCK	2-87
REALITY	2-87
REALITY.FORMAT	2-87
RECORD	2-88
RECORD.SIZE	2-88
–REJECT	2-88
REMOVING	2-89
REPORTING	2-89
REQUIRE.SELECT	2-89
RESTORE	2-90
RESTOREDATA	2-90
RETAIN	2-90
RUNNING.	2-90
SAID	2-90
SAMPLE	2-91
SAMPLED	2-91
SAVEDATA	2-91
SAVING	2-91
SCHEMAS	2-92
SELECT.BUFFER	2-92
SELECT.ONLY	2-92
SEQ.NUM.	2-93
SET	2-93
SHOW	2-93
SINGLE.VALUE.	2-93
SOME	2-93
SPLIT.LOAD	2-94
SPOKEN	2-94
SPOOL	2-94
-SPOOL	2-94
SQL	2-94
SQUAWK	2-95
STATISTICS	2-95
STATS	2-96
SUPP	2-96
–SUPPRESS	2-96
SUSPENDED.	2-96
SYSTEM	2-97
TAPE	2-97

TEMPL	2-97
TEST	2-97
THAN.	2-97
THE	2-97
THEN.	2-98
TIME	2-98
TO	2-98
TOTAL	2-99
TRANSPORT	2-99
TRAP	2-100
TRUNCATE.	2-100
TTY.ON	2-100
TYPE	2-101
-UNICODE	2-101
UNION	2-101
UNIQUE	2-101
UNLIKE	2-102
UP	2-102
UPDATING	2-102
USER.	2-102
USERS	2-102
USING	2-102
UTF8	2-103
VERIFIELD.	2-103
VERIFILE	2-103
VERIFY	2-104
VERT	2-105
VERTICALLY	2-105
WHEN	2-105
WITH.	2-106
WITHIN	2-108
-XREF	2-109

Chapter 3 User Records

Alphabetical Summary of User Records	3-3
INTR.KEY	3-4
QUIT.KEY.	3-6
RELLEVEL	3-8
STACKWRITE	3-9

SUSP.KEY	3-10

Chapter 4 **Local and System Files**

&COMO&	4-3
&DEVICE&	4-4
&ED&	4-6
&HOLD&	4-8
&MAP&	4-9
&PARTFILES&	4-10
&PH&	4-11
&SAVEDLISTS&	4-12
&TEMP&	4-13
&UFD&	4-14
&UNICODE.FILE&	4-15
APP.PROGS	4-16
BLTRS	4-17
DICT.DICT	4-18
DICT.PICK	4-19
NEWACC	4-20
REVISE.PROCESSES	4-21
STAT.FILE	4-22
SYS.HELP	4-23
SYS.MESSAGE	4-24
UNIVERSE.MENU.FILE	4-25
UNIVERSE.STAT.FILE	4-26
UNIVERSE.VOCLIB	4-27
USER.HELP	4-28

Appendix A **UniVerse Commands Quick Reference**

Redirecting Output	A-4
Paragraph Commands	A-5
RetrieVe Commands	A-6
Creating and Manipulating Select Lists	A-8
Managing UniVerse Files	A-10
Managing Records	A-11
Managing UniVerse File Structures.	A-12
Managing Secondary Indexes	A-14
Managing BASIC Programs	A-15
Arithmetic Commands.	A-16
Account Commands	A-17
VOC File Commands	A-19
Managing Menus	A-21

Printer Commands	A-22
Terminal Commands	A-24
Tape Commands	A-25
Managing the System	A-26
Managing Locks	A-29
Managing Transaction Logging	A-30
NLS Commands	A-31

Preface

This manual is a guide to the resources available in the UniVerse environment. It presents the UniVerse commands and keywords in an alphabetical listing for easy reference.

Organization of This Manual

This manual contains the following:

Chapter 1, “[UniVerse Commands](#),” is an alphabetical listing and description of UniVerse commands.

Chapter 2, “[UniVerse Keywords](#),” is an alphabetical listing and description of UniVerse keywords that modify UniVerse commands.

Chapter 3, “[User Records](#),” is an alphabetical listing and description of UniVerse user records defined in the VOC file.

Chapter 4, “[Local and System Files](#),” is an alphabetical listing and description of local and system files used by UniVerse processes.

Appendix A, “[UniVerse Commands Quick Reference](#),” is a quick reference for UniVerse commands grouped according to use.

Documentation Conventions

This manual uses the following conventions:

Convention	Usage
Bold	In syntax, bold indicates commands, function names, and options. In text, bold indicates keys to press, function names, menu selections, and MS-DOS commands.
UPPERCASE	In syntax, uppercase indicates UniVerse commands, keywords, and options; BASIC statements and functions; and SQL statements and keywords. In text, uppercase also indicates UniVerse identifiers such as filenames, account names, schema names, and Windows file names and paths.
<i>Italic</i>	In syntax, italic indicates information that you supply. In text, italic also indicates UNIX commands and options, file names, and paths.
Courier	Courier indicates examples of source code and system output.
Courier Bold	In examples, courier bold indicates characters that the user types or keys the user presses (for example, <Return>).
[]	Brackets enclose optional items. Do not type the brackets unless indicated.
{ }	Braces enclose nonoptional items from which you must select at least one. Do not type the braces.
itemA itemB	A vertical bar separating items indicates that you can choose only one item. Do not type the vertical bar.
...	Three periods indicate that more of the same type of item can optionally follow.
%	A right arrow between menu options indicates you should choose each option in sequence. For example, “Choose File % Exit” means you should choose File from the menu bar, then choose Exit from the File pull-down menu.
␣	Item mark. For example, the item mark (␣) in the following string delimits elements 1 and 2, and elements 3 and 4: 1␣2F3␣4V5

Convention	Usage
F	Field mark. For example, the field mark (F) in the following string delimits elements FLD1 and VAL1: FLD1 F VAL1 V SUBV1 S SUBV2
V	Value mark. For example, the value mark (V) in the following string delimits elements VAL1 and SUBV1: FLD1 F VAL1 V SUBV1 S SUBV2
S	Subvalue mark. For example, the subvalue mark (S) in the following string delimits elements SUBV1 and SUBV2: FLD1 F VAL1 V SUBV1 S SUBV2
T	Text mark. For example, the text mark (T) in the following string delimits elements 4 and 5: 1 F 2 S 3 V 4 T 5

Documentation Conventions (Continued)

The following conventions are also used:

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

UniVerse Documentation

UniVerse documentation includes the following:

UniVerse Installation Guide: Contains instructions for installing UniVerse 10.3.

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

UniVerse BASIC: Contains comprehensive information about the UniVerse BASIC language. It includes reference pages for all BASIC statements and functions. It is for experienced programmers.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Related Documentation

The following documentation is also available:

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

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

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

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

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

Client API Documentation

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

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

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

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

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

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

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

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

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

UniVerse Commands

.A	1-11
.C	1-13
.D	1-16
.I.	1-18
.L	1-19
.R	1-21
.S.	1-23
.U	1-25
.X	1-26
.?.	1-28
*	1-29
<<...>>	1-31
ABORT	1-34
ABORT.LOGIN	1-35
ACCOUNT.FILE.STATS	1-36
ACTIVATE.ENCRIPTION.KEY	1-38
ACTLIST	1-39
ANALYZE.FILE.	1-41
analyze.shm	1-45
ANALYZE.SHM.	1-55
ASSIGN	1-60
AUTOLOGOUT	1-63
AVAIL	1-64
BASIC	1-66
BELL	1-69
BLOCK.PRINT	1-70
BLOCK.TERM	1-72

Bea

CREATE.LFILE	. 1-171
CS	. 1-173
CSH	. 1-174
CT	. 1-176
DATA	. 1-178
DATE	. 1-180
DATE.FORMAT	. 1-181
DEACTIVATE.ENCRIPTION.KEY	. 1-184
DEACTLIST	. 1-185
DECATALOG	. 1-186
DECRYPT.FILE	. 1-187
DEFINE.DF	. 1-191
DEL.RFILE	. 1-197
DELETE	. 1-199
DELETE.CATALOG	. 1-201
DELETE.ENCRIPTION.KEY	. 1-203
DELETE.FILE	. 1-205
DELETE.INDEX	. 1-208
DELETE.LFILE	. 1-209
DELETE.LIST	. 1-211
DISABLE.DECRYPTION	. 1-212
DISABLE.INDEX	. 1-213
DISPLAY	. 1-215
DIVERT.OUT	. 1-216
DLIST.	. 1-219
DOS	. 1-221
ED	. 1-222
EDIT.CONFIG	. 1-223
EDIT.LIST	. 1-226
ENABLE.DECRYPTION	. 1-227
ENABLE.INDEX	. 1-228
ENABLE.RECOVERY	. 1-229
ENCRYPT.FILE	. 1-231
ENVIRONMENT or ENV	. 1-236
ESEARCH	. 1-238
EXCHANGE	. 1-239

FANCY.FORMAT	1-240
FILE.STAT.	1-241
FILE.USAGE	1-243
FILE.USAGE.CLEAR	1-246
FILE.USAGE.OFF	1-247
FORM.LIST	1-248
FORMAT	1-250
FORMAT.CONV.	1-252
GET.FILE.MAP	1-258
GET.LIST	1-260
GET.LOCALE	1-262
GET.STACK	1-264
GET.TERM.TYPE	1-265
GO	1-268
GRANT.ENCRYPTION.KEY.	1-270
GROUP.STAT.	1-272
GROUP.STAT.DETAIL.	1-274
HASH.AID	1-276
HASH.HELP	1-280
HASH.HELP.DETAIL	1-282
HASH.TEST	1-284
HASH.TEST.DETAIL	1-286
HELP	1-289
HUSH	1-292
IAM	1-293
IF.	1-295
INITIALIZE.CATALOG	1-297
INITIALIZE.DEMO.	1-299
JOBS	1-300
LIMIT	1-301
LIST.	1-302
LIST.DF	1-306
LIST.DIFF	1-307
LIST.ENCRYPTION.FILE.	1-310
LIST.ENCRYPTION.KEY.	1-311
LIST.ENCRYPTION.WALLET	1-312

LIST.FILE.STATS	. 1-313
LIST.INDEX	. 1-315
LIST.INTER.	. 1-319
LIST.ITEM	. 1-322
LIST.LABEL	. 1-325
LIST.LOCALES	. 1-330
LIST.LOCKS	. 1-332
LIST.MAPS	. 1-334
LIST.READU	. 1-336
LIST.SICA	. 1-341
LIST.UNION	. 1-344
LISTME	. 1-347
LISTU.	. 1-349
LIST	. 1-351
LOCK.	. 1-353
LOG.RESTORE.	. 1-355
LOG.SAVE	. 1-357
LOGIN	. 1-359
LOGON	. 1-361
LOGOUT.	. 1-362
LOGTO	. 1-364
LOGTO.ABORT	. 1-365
LOOP	. 1-366
MAIL	. 1-368
MAKE.DEMO.FILES.	. 1-371
MAKE.DEMO.TABLES	. 1-372
MAKE.MAP.FILE	. 1-373
MAP	. 1-375
MASTER.	. 1-376
MENU.DOC.	. 1-377
MENU.PIX	. 1-378
MENUS	. 1-379
MERGE.LIST	. 1-380
MESSAGE	. 1-383
MKFILELIST	. 1-385
MOTIF	. 1-387

NLS.ADMIN	1-390
NLS.UPDATE.ACCOUNT	1-391
NOTIFY	1-393
NSELECT	1-394
OFF	1-396
ON.ABORT	1-398
ON.EXIT	1-399
P.ATT	1-400
P.DET	1-401
PAGE.MESSAGE	1-402
PASSWD	1-403
PHANTOM	1-404
PORT.STATUS	1-407
PRIME	1-411
PRINT.ADMIN	1-412
PTERM (UNIX)	1-413
PTERM (Windows Platforms).	1-430
PTIME	1-438
QSELECT	1-439
QUIT	1-441
RADIX	1-442
RAID	1-446
RAID.GUI	1-449
REBUILD.DF.	1-452
RECORD	1-456
RECOVERY.CHECKPOINT	1-460
RECOVERY.CONSISTENT	1-462
REFORMAT	1-463
RELEASE	1-468
RELEASE.LFILE	1-469
REMOVE.DEMO.FILES	1-471
REMOVE.DEMO.TABLES	1-472
REPEAT	1-473
RESIZE.	1-474
RESTORE.LOCALE	1-479

REVISE	. 1-480
REVOKE.ENCRIPTION.KEY	. 1-482
RUN	. 1-483
SAVE.LIST	. 1-485
SAVE.LOCALE	. 1-487
SAVE.STACK	. 1-488
SEARCH	. 1-489
SELECT	. 1-492
SEMAPHORE.STATUS	. 1-495
SET.FILE	. 1-497
SET.FILE.MAP	. 1-498
SET.GCI.MAP	. 1-501
SET.INDEX	. 1-503
SET.LOCALE	. 1-507
SET.LOG.ATTR	. 1-509
SET.REMOTE.ID	. 1-511
SET.SEQ.MAP	. 1-513
SET.SQL	. 1-515
SET.TERM.TYPE	. 1-518
SETFILE	. 1-521
SETPTR (UNIX)	. 1-522
SETPTR (Windows Platforms)	. 1-530
SETPTR.DEFAULT	. 1-539
SETUP.DEMO.SCHEMA	. 1-540
SH	. 1-541
SHUTDOWN.RECOVERY	. 1-543
SLEEP	. 1-545
SORT	. 1-547
SORT.ITEM	. 1-551
SORT.LABEL	. 1-554
SP.ASSIGN (UNIX)	. 1-559
SP.ASSIGN (Windows Platforms)	. 1-561
SP.EDIT (UNIX)	. 1-562
SP.EDIT (Windows Platforms)	. 1-567
SP.TAPE	. 1-571
SPOOL	. 1-573

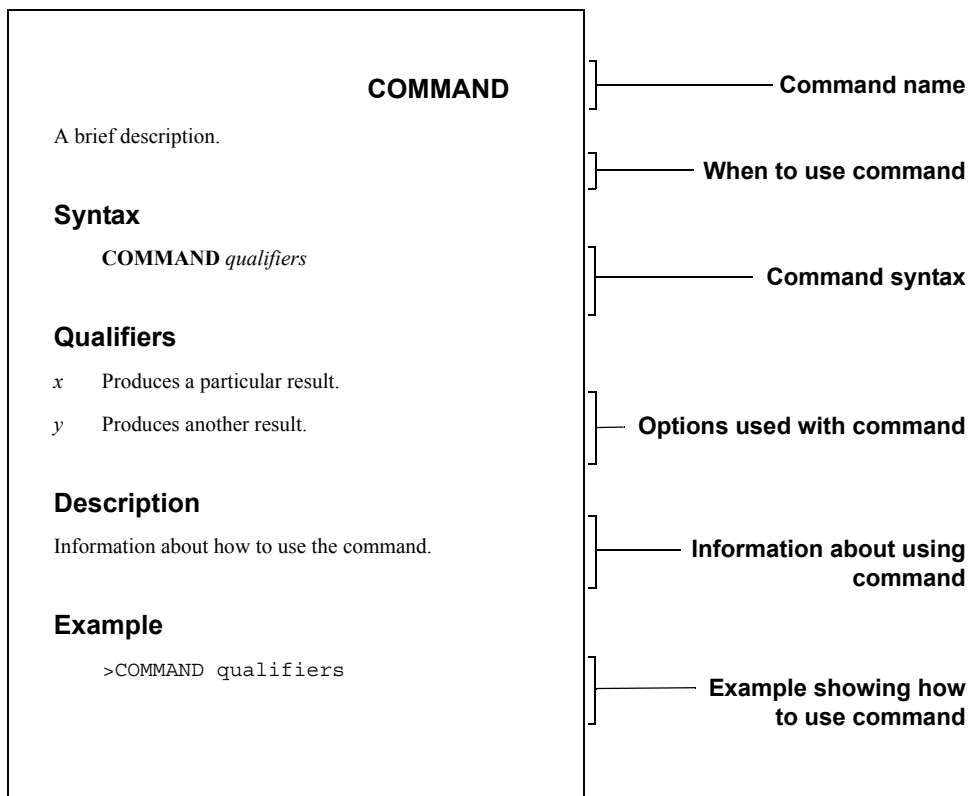
SPOOL (Windows Platforms)	1-576
SREFORMAT	1-578
SSELECT	1-583
STAT	1-586
STATUS	1-588
SUM.	1-589
SUSPEND.FILES	1-591
SUSPEND.RECOVERY	1-593
T.ATT	1-594
T.BCK	1-596
T.DET	1-597
T.DUMP	1-598
T.EOD	1-601
T.FWD	1-602
T.LOAD	1-603
T.RDLBL	1-605
T.READ	1-607
T.REW	1-609
T.SPACE	1-610
T.UNLOAD	1-611
T.WEOF	1-612
T.WTLBL	1-613
TANDEM	1-616
TERM	1-619
TIME	1-622
UMASK	1-623
UNASSIGN	1-626
UNICODE.FILE	1-627
UNLOCK	1-629
UNLOCK SEMAPHORE	1-631
UPDATE.ACCOUNT	1-633
UPDATE.INDEX.	1-635
usa	1-637
usd	1-640
USERS	1-642
usm	1-643

usp	. 1-645
uv	. 1-647
UV.LOGIN	. 1-649
UV.VI	. 1-651
uvbackup (UNIX)	. 1-653
uvbackup (Windows Platforms)	. 1-657
uvdlockd	. 1-660
uvfixfile	. 1-662
UVFIXFILE	. 1-669
UVHELP	. 1-677
uvlictool	. 1-679
UVPROMPT	. 1-680
uvrestore (UNIX)	. 1-681
uvrestore (Windows Platforms)	. 1-685
VCATALOG	. 1-689
VERIFY.DF	. 1-691
VERIFY.SQL	. 1-693
VI	. 1-699
VLIST	. 1-701
VVOC	. 1-703
WHO	. 1-707

This chapter describes every command in the IDEAL UniVerse flavor VOC file. The commands are arranged in alphabetical order.

The term UniVerse Administrator refers to a UNIX *root* or *uvadm* user, or to a Windows Administrator.

The following sample shows a typical command reference page.



.A

Use .A (Append) to add text to the end of a sentence in the sentence stack.

Syntax

`.A [ n ] text`

Parameters

Parameter	Description
<i>n</i>	The line number of the sentence in the stack to which you want to append text. If you do not include the line number, <i>text</i> is appended to sentence 1.
<i>text</i>	Any text that you want to add to a sentence. Text is appended to the specified sentence immediately after the last nonblank in the sentence. If you want the text separated from the original sentence by a blank, put at least two blanks before <i>text</i> .

.A Parameters

Description

The .A command does not execute the newly created sentence. It just changes the sentence in the sentence stack. You can execute the changed sentence, or you can save it using sentence stack commands.

Example

Sentence 1 in the stack is as follows:

```
01 LIST PAYABLES WITH PYMT.DATE >= 1/1/95
```

Add the following text. Notice that there are two spaces before AND.

```
>.A   AND AMT.PD < 10
```

The new sentence now looks like this:

```
01 LIST PAYABLES WITH PYMT.DATE >= 1/1/95 AND AMT.PD < 10
```

.C

Use **.C** (Change) to change a sentence in the sentence stack. This command is useful if you make a mistake typing the original sentence or if you want to execute a sentence that is only partially different from the original sentence.

Syntax

.C [*n*] /*string1/string2* [/ [*G*]]

Parameters

Parameter	Description																												
<i>n</i>	Specifies the line number of the sentence in the stack where text is to be changed. If you do not include the line number, the text in sentence 1 is changed.																												
<i>string1</i>	The sequence of characters you want to change in the original sentence.																												
<i>string2</i>	The string of characters that you want to replace <i>string1</i> . If <i>string2</i> is an empty string, <i>string1</i> is deleted from the original sentence.																												
/	A delimiter put before <i>string1</i> , <i>string2</i> , and the optional G. The delimiter can be any of the following, but you must be consistent within a single command (see the examples): <table><tr><td>!</td><td>@</td><td>#</td><td>\$</td><td>%</td><td>&</td><td>*</td></tr><tr><td>/</td><td>\</td><td>:</td><td>=</td><td>+</td><td>–</td><td>?</td></tr><tr><td>(</td><td>)</td><td>{</td><td>}</td><td>[</td><td>]</td><td></td></tr><tr><td>`</td><td>'</td><td>.</td><td> </td><td>"</td><td>,</td><td></td></tr></table>	!	@	#	\$	%	&	*	/	\	:	=	+	–	?	(	)	{	}	[	]		`	'	.		"	,	
!	@	#	\$	%	&	*																							
/	\	:	=	+	–	?																							
(	)	{	}	[	]																								
`	'	.		"	,																								
G	Changes every occurrence of <i>string1</i> in the sentence to <i>string2</i> . If you do not specify G, only the first occurrence of <i>string1</i> is changed.																												

.C Parameters

Description

The .C command does not execute the newly created sentence. You execute or save the changed sentence using sentence stack commands.

Example

Sentence 1 in the stack is as follows:

```
01 LIST PAYABLES WITH PYMT.DATE >= 1/1/95
```

To change the month to November, enter the following:

>.C:1/1/95:11/1/95:

Notice that colons delimit the old and new dates because the dates contain slashes.
The new sentence is as follows:

01 LIST PAYABLES WITH PYMT.DATE >= 11/1/95

.D

Use .D (Delete) to delete a sentence from the sentence stack or to delete a sentence or paragraph from the VOC file.

Syntax

.D [*n* | *name*]

Parameters

Parameter	Description
<i>n</i>	Specifies the line number of the sentence in the stack is to be deleted. If you do not include the line number, sentence 1 is deleted.
<i>name</i>	The name of the VOC file entry for the sentence or paragraph you want to delete. If the VOC file entry is not a sentence or paragraph name, the .D command issues an error message and does not delete the entry.

.D Parameters

Description

Before deleting a stored sentence or paragraph, you can examine the contents of the sentence or paragraph using the .L (List) command.

Example

There are two sentences in the stack:

```
>.L
02 LIST PAYABLES WITH PYMT.DATE >= 1/1/95
01 LIST PAYABLES WITH PYMT.DATE >= 12/1/94
```

Delete sentence 1 by entering the following:

```
>.D
```

The stack now contains this sentence:

```
01 LIST PAYABLES WITH PYMT.DATE >= 1/1/95
```

.I

Use .I (Insert) to insert a new sentence into the sentence stack. This feature is useful for building paragraphs because you can add a sentence to those already in the sentence stack.

Syntax

.I [*n*] *text*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>n</i>	Specifies the number of the new sentence in the sentence stack. The sentence numbers greater than or equal to <i>n</i> increase by one. If you do not include the sentence number, the sentence is inserted as sentence 1.
<i>text</i>	The sentence you want to insert in the sentence stack.

.I Parameters

Example

The existing stack contains the following:

```
02 LIST PAYABLES WITH PYMT.DATE >= 1/1/95
01 LIST PAYABLES WITH PYMT.DATE >= 12/1/94
```

Insert the following:

```
>.I2 LIST PAYABLES WITH AMT.DUE > 100
```

The new stack looks like this:

```
03 LIST PAYABLES WITH PYMT.DATE >= 1/1/95
02 LIST PAYABLES WITH AMT.DUE > 100
01 LIST PAYABLES WITH PYMT.DATE >= 12/1/94
```

.L

Use .L (List) to list the sentences in the sentence stack or to list the contents of a VOC file entry.

Syntax

.L [*n* | *name*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>n</i>	The number of sentences to display. The number can be from 1 through 99.
<i>name</i>	The name of any VOC file entry to be displayed. This command is useful for verifying the contents of a sentence or paragraph before executing it.

.L Parameters

Description

If you do not specify *n*, sentences 1 through 20 are displayed. If you choose a number greater than 22, the first 23 sentences are displayed followed by the message:

Press any key to continue...

When you press any key but Q, the next 23 sentences are displayed. This continues until all the sentences you requested have been displayed, or until you press Q to quit.

Examples

To list the three most recent sentences, enter the following:

```
>.L3
03 LIST PAYABLES WITH PYMT.DATE >= 1/1/95
02 LIST PAYABLES WITH AMT.DUE > 100
01 LIST PAYABLES WITH PYMT.DATE >= 12/1/94
```

To list the contents of a stored sentence called MONTH, enter the following:

>.L MONTH

MONTH

001 S

002 LIST PAYABLES WITH PYMT.DATE >= 12/1/94

.R

Use .R (Recall) to recall a sentence in the sentence stack or a sentence or paragraph stored in the VOC file.

Syntax

.R [*n* | *name*]

Parameters

Parameter	Description
<i>n</i>	Specifies the line number of the sentence to recall. The number can be from 1 through 99. If you do not include a number, sentence 1 is redisplayed.
<i>name</i>	The name of the sentence or paragraph. The named sentence or paragraph is inserted in the sentence stack.

.R Parameters

Description

When you recall a sentence, you copy it to sentence 1 in the sentence stack. When you recall a paragraph, you copy the sentences in the paragraph to the first *n* sentences in the sentence stack, where *n* is the number of sentences in the paragraph.

The .R command does not execute the newly created sentence. It just changes the sentence in the sentence stack. You can execute the changed sentence or you can save it using sentence stack commands.

If you want to change a stored sentence or paragraph before executing it, use .R to insert the sentence or paragraph into the sentence stack. Once the sentence or paragraph is in the sentence stack, you can use any of the sentence stack commands to change it before executing it.

Examples

The existing stack looks like this:

```
03 LIST PAYABLES WITH PYMT.DATE >= 1/1/95
02 LIST PAYABLES WITH AMT.DUE > 100
01 LIST PAYABLES WITH PYMT.DATE >= 12/1/94
```

To recall sentence 3, enter the following:

```
>.R3
```

The stack looks like this:

```
04 LIST PAYABLES WITH PYMT.DATE >= 1/1/95
03 LIST PAYABLES WITH AMT.DUE > 100
02 LIST PAYABLES WITH PYMT.DATE >= 12/1/94
01 LIST PAYABLES WITH PYMT.DATE >= 1/1/95
```

To recall a stored sentence named LISTME, enter the following:

```
>.R LISTME
```

The stack now looks like this:

```
05 LIST PAYABLES WITH PYMT.DATE >= 1/1/95
04 LIST PAYABLES WITH AMT.DUE > 100
03 LIST PAYABLES WITH PYMT.DATE >= 12/1/94
02 LIST PAYABLES WITH PYMT.DATE >= 1/1/95
01 STATUS ME
```

.S

Use .S (Save) to save a sentence or paragraph as an entry in the VOC file.

Syntax

.S name [start [end]]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>name</i>	The name of the entry in the VOC file. When you later execute or recall the sentence or paragraph, use this name.
<i>start</i>	The number of the first sentence you want to save. If you do not specify <i>start</i> , sentence 1 is saved.
<i>end</i>	The number of the last sentence you want to save. When you specify <i>end</i> , all sentences from <i>start</i> through <i>end</i> are saved as a paragraph. If <i>end</i> is the same as <i>start</i> , just that sentence is saved.

.S Parameters

Description

You can specify *start* and *end* either in ascending or in descending order. The .S command always saves a range of sentences in descending order from oldest to newest.

If a VOC entry with the same name and a type of sentence or paragraph already exists, the .S command displays this prompt:

Enter Option: O) Overwrite, N) Choose a New Name, A) Abort?

Enter the letter **O** to overwrite the VOC entry, enter **N** to rename the new VOC entry, or enter **A** to abort the .S command. If the VOC entry exists and is a type other than sentence or paragraph, the .S command issues an error message and prompts you for a new name.

Example

The existing stack looks like this:

```
05 LIST PAYABLES WITH PYMT.DATE >= 1/1/95
04 SELECT PAYABLES WITH AMT.DUE > 100
03 LIST PAYABLES WITH PYMT.DATE >= 12/1/94
02 LIST PAYABLES WITH PYMT.DATE >= 1/1/95
01 LIST PAYABLES WITH AMT.DUE > 10 NAME DUE.DATE
```

Save lines 4 and 3 as a paragraph called PAYMENTS.DUE by entering the following:

```
>.S PAYMENTS.DUE 4 3
```

List the paragraph PAYMENTS.DUE by entering the following:

```
>.L PAYMENTS.DUE
```

This displays the following:

```
PAYMENTS.DUE
001 PA saved on 10:34:12 FEB 23 1995 by JONES
002 SELECT PAYABLES WITH AMT.DUE > 100
003 LIST PAYABLES WITH PYMT.DATE >= 12/1/94
```

The order of the sentences is the same in the sentence stack and the paragraph, but the numbering is different. The paragraph numbering indicates the order in which the paragraph executes the commands.

.U

Use .U (Uppercase) to convert a sentence from lowercase to uppercase.

Syntax

.U [*n*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>n</i>	Specifies the number of the sentence to be converted to uppercase. If you do not specify <i>n</i> , sentence 1 is converted.

.U Parameters

Example

Sentence 4 looks like this:

```
04 select payables with amt.due > 100
```

Convert sentence 4 by entering the following:

```
> .U4
```

Now it looks like this:

```
04 SELECT PAYABLES WITH AMT.DUE > 100
```

.X

Use **.X** (eXecute) to execute a sentence in the sentence stack.

Syntax

.X [*n*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>n</i>	Specifies the number of the sentence to be executed. If you do not specify <i>n</i> , sentence 1 is executed.

.X Parameters

Description

When you execute a sentence other than sentence 1, the command processor first copies it to sentence 1 and then executes it.

Example

The existing stack looks like this:

```
04 SELECT PAYABLES WITH AMT.DUE > 100
03 LIST PAYABLES WITH PYMT.DATE >= 12/1/94
02 LIST PAYABLES WITH PYMT.DATE >= 1/1/95
01 LIST PAYABLES WITH AMT.DUE > 10 NAME DUE.DATE
```

Execute sentence 3 by entering the following:

```
> .X3
```

Now the stack looks like this:

```
05 SELECT PAYABLES WITH AMT.DUE > 100
04 LIST PAYABLES WITH PYMT.DATE >= 12/1/94
03 LIST PAYABLES WITH PYMT.DATE >= 1/1/95
02 LIST PAYABLES WITH AMT.DUE > 10 NAME DUE.DATE
01 LIST PAYABLES WITH PYMT.DATE >= 12/1/94
```

.?

Use ? (question mark) at the end of a sentence to save the sentence in the sentence stack without executing it.

Syntax

?

Description

If you are typing a sentence and you notice you made a typing mistake before completing the sentence, you can save the sentence in the stack without executing it. Because the sentence has been saved in the sentence stack, you do not have to retype the entire sentence. Instead, you can correct the mistake using .C and then execute the sentence.

You can also use ? to enter paragraph control statements into the sentence stack before using a .S command to save the paragraph.

Example

Type ? at the end of a sentence that contains an error:

```
>SORT PAYABLES BY VENDOR BYSTATE?
```

Change the sentence that has been saved in the sentence stack:

```
>.C/Ys/Y S/  
01 SORT PAYABLES BY VENDOR BY STATE
```

Execute the corrected sentence:

```
>.X
```

Use * (asterisk) in a paragraph to indicate that the following text is a comment.

Syntax

* [*comment*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>comment</i>	The text of the comment.

*** Parameters**

Description

The comment does not affect the operation of the sentences in the paragraph. A comment line must contain at least one space character after the asterisk (*) to distinguish it from a cataloged BASIC program name.

You can enter comments at the UniVerse command prompt (>) as well as in paragraphs, which is helpful if you are using a COMO file and want to comment on something for later reference. For information about COMO files, see the [COMO](#) command.

You can include inline prompts in comment lines. Inline prompts in comment lines are processed and displayed as usual. For information about inline prompts, see the [<<...>>](#) command.

Example

This example shows a paragraph called LIST.PAYMENTS. Lines 2, 4, and 6 are comments.

```
LIST.PAYMENTS
0001: PA
0002: * List the balances.due greater than 10
0003: BALANCES
0004: * List the most recent payment for all vendors
0005: PAYMENTS
0006: * Print a list of payments and balances
0007: PAY.REPORT
```

<<...>>

<< ... >> denotes inline prompting. Use inline prompting in a paragraph to display a prompt and to request an input value when the sentence is executed. The command processor executes the sentence as if the input value had been entered instead of the inline prompt specification. The input value can be requested from the terminal, extracted from a UniVerse file, or supplied by DATA statements in a BASIC program.

Syntax

<< [*control*,] ... *text* [,*option*] >>

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>control</i>	Specifies the characteristics of the prompt. Possible control options are as follows: A Always prompts when the sentence containing the control option is executed. If this option is not specified, the input value from a previous execution of this prompt is used. C <i>n</i> Uses word <i>n</i> from the command line as the input value. (The verb is word 1.) F(<i>fn</i> , <i>rn</i> [, <i>fm</i> [, <i>vm</i> [, <i>sm</i>]]]) Finds the input value from a record (<i>rn</i>) in a file (<i>fn</i>). You must specify the file and the record. Separate them with a comma (no spaces allowed). <i>vm</i> extracts a value, and <i>sm</i> extracts a subvalue from the field (<i>fm</i>). Separate values with commas (no spaces allowed). In Takes word <i>n</i> from the command line, but prompts if word <i>n</i> was not entered.

<< **Parameters**

Parameter	Description
P	Saves the input value from an inline prompt. The input value is used for all inline prompts with the same prompt text, until the input value is overwritten by a prompt with the same prompt text and with a control option of A, C, I, or S, or until control returns to the UniVerse prompt. P saves the input value from an inline prompt in the current paragraph or in other paragraphs.
R	Repeats the prompt until the user presses ENTER.
R(<i>string</i>)	Repeats the prompt until the user presses ENTER, and inserts <i>string</i> between each entry.
Sn	Takes word <i>n</i> from the command but uses the most recent command entered at the UniVerse level to execute the paragraph, rather than an argument in the paragraph. This is useful when paragraphs are nested.
@(CLR)	Clears the screen.
@(BELL)	Rings a bell.
@(TOF)	Issues top of form.
@(<i>col, row</i>)	Prompts at this column and row number on the terminal.
<i>text</i>	The prompt to be displayed.
<i>option</i>	Any conversion or pattern match. Conversions are the same as the BASIC ICONV function . Patterns are the same as those used with the BASIC MATCH operator. A conversion must be in parentheses. The conversion only verifies the format of the data entered. It does not convert the data. If the data does not match the conversion, the user is prompted to reenter it.

<< Parameters (Continued)

Description

Once a value has been entered for a particular prompt, the prompt continues to have that value until a **CLEARPROMPTS** command is executed if control option A is not specified. CLEARPROMPTS clears all of the values that have been entered for inline prompts.

You can enclose prompts within prompts.

You cannot use an inline prompt as the verb of a sentence.

Examples

Enter a value:

```
<<ENTER NAME>>
```

Enter a value and check that it is a date:

```
<<ENTER DATE, (D) >>
```

Clear the screen and prompt for AMOUNT at column 10, row 5:

```
<<@(CLR) ,@(10,5) ,AMOUNT>>
```

The following paragraph prompts for a new value each time the loop is repeated. If you omit the **CLEARPROMPTS** statement, the loop repeats endlessly using the value for <<ENTER MEMBERSHIP.NBR>> entered at the initial prompt.

```
LIST.MEMBERS
0001: PA
0002: LOOP
0003: CLEARPROMPTS
0004: IF <<ENTER MEMBERSHIP.NBR>> EQ 'END' GO END:
0005: LIST SUN.MEMBER WITH @ID EQ <<ENTER MEMBERSHIP.NBR>>
0006: REPEAT
0007: END:
```

ABORT

Use ABORT in a paragraph or in a BASIC EXECUTE statement to abort the current process. This is useful in conditional clauses to stop a process when certain conditions exist.

Syntax

ABORT

Example

The following paragraph, TEST, aborts when nothing is entered at the MEM.ID prompt:

```
TEST
0001: PA
0002: LOOP
0003: CLEARPROMPTS
0004: IF <<MEM.ID>> = ' ' THEN ABORT
0005: LIST SUN.MEMBER <<MEM.ID>>
0006: REPEAT
```

To execute it, enter **TEST**:

```
1  >TEST
MEM.ID=7505
LIST SUN.MEMBER 7505 11:08:57am 20 Oct 1995 PAGE 1
MEMBER ID.      FIRST NAME    LAST NAME.      YEAR JOINED
INTERESTS.....

7505            HARRY          EDWARDS         1978
SURFING

DOMINOES

VOLLEYBALL

DIVING

1 records listed.
MEM.ID=
```

ABORT.LOGIN

Use ABORT.LOGIN to abort the current process and rerun the LOGIN paragraph. This reestablishes the defaults that are set up in the LOGIN paragraph.

Syntax

ABORT.LOGIN

Example

The following BASIC program executes an ABORT.LOGIN statement to restart after an error:

```
OPEN ' ', 'PAYABLES' TO FILE.PAYABLES ELSE
  PRINT 'CANNOT OPEN PAYABLES FILE'
  PRINT 'PAYABLES PROCESSING CANNOT BE COMPLETED'
  EXECUTE 'ABORT.LOGIN'
END
PRINT 'BEGINNING PAYABLES PROCESSING'
.
.
.
PRINT 'PAYABLES PROCESSING COMPLETED'
STOP
END
```

ACCOUNT.FILE.STATS

Use ACCOUNT.FILE.STATS to gather file statistics on the current state of selected files. To gather statistics on a file, you must have permission to read it.

Syntax

ACCOUNT.FILE.STATS [*filenames* | ALL | *] [LOCAL]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>filenames</i>	Specifies the files whose statistics you want to gather. If a select list of file names is active, you do not need to specify <i>filenames</i> .
ALL	Specifies that statistics for all files in the account are to be gathered.
*	Same as ALL.
LOCAL	Specifies that file statistics are to be stored in a local STAT.FILE file. This file is created if it does not exist. If you do not specify LOCAL, the STAT.FILE file in the UV account is used. If the STAT.FILE does not exist, you must have write privileges to the current directory to use the LOCAL keyword.

ACCOUNT.FILE.STATS

Description

ACCOUNT.FILE.STATS writes records to the STAT.FILE file. Each file receives a unique key in the following format:

*date * time * seq*

Format	Description
<i>date</i>	The system date in internal format.
<i>time</i>	The system time in internal format.
<i>seq</i>	A sequential four-digit number starting with 000.
<i>*</i>	(Asterisk) The delimiter character.

ACCOUNT.FILE.STATS Key Format

*date * time * seq*

Statistics are gathered for a file according to its file type. If the file is nonhashed (types 1, 19, and 25), the file is opened, all records are selected and read, and statistics are gathered. Nonhashed files produce the least data. If the file is a dynamic file (type 30), the [ANALYZE.FILE](#) command is issued, and the relevant statistics are written to the STAT.FILE file. If the file is a hashed file (types 2 through 18), a [FILE.STAT](#) command is issued, and the relevant statistics are written to STAT.FILE.

ACCOUNT.FILE.STATS does not clear the STAT.FILE file, nor does it display output other than error messages at the terminal. To list the file statistics in the FILE.STAT file, use the [LIST.FILE.STATS](#) command.

The time it takes to process files in an account depends on the actual number and size of the files being examined.

ACTIVATE.ENCRYPTION.KEY

Use the ACTIVATE.ENCRYPTION.KEY command to activate a key. It is necessary to activate a key if you want to supply a password for key protection.

Syntax

ACTIVATE.ENCRYPTION.KEY *key.id password* [ON *<hostname>*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>key.id</i>	The key ID to activate.
<i>password</i>	The password corresponding to <i>key.id</i> .
ON <i>hostname</i>	The name of the remote host on which you want to activate the encryption key.

ACTIVATE.ENCRYPTION.KEY Parameters

Note: You can activate only keys with password protection with this command. Keys that do not have password protection are automatically activated. Also, you can activate only keys to which you are granted access.



ACTLIST

Use ACTLIST to activate files for transaction logging. You must be a UniVerse Administrator logged in to the UV account to use ACTLIST.

Syntax

ACTLIST *listname*

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>listname</i>	The record ID of the select list created and saved in the &SAVEDLISTS& file by the MKFILELIST command.

ACTLIST Parameters

Description

ACTLIST activates the data file, the file dictionary, and any secondary indexes for each file in the select list. ACTLIST cannot activate a type 1 or a type 19 file. It displays an error message for each type 1 or type 19 file in the select list.

Before you execute ACTLIST, the index called FILE in the UV.TRANS file must be up-to-date. If it is not, a message reminds you to rebuild the index with the [BUILD.INDEX](#) command. If for some reason the FILE index does not exist, a message tells you to create the index with the [CREATE.INDEX](#) command, and to build the index with BUILD.INDEX.

Any errors occurring during file activation appear on the terminal screen. ACTLIST does not report potential data inconsistencies due to reactivation, nor does it report inconsistencies between the header of a file and the status of the same file as recorded in UV.TRANS.

Example

This example activates all files listed in the saved list INV.FILE.LIST:

```
>ACTLIST INV.FILE.LIST
```

ANALYZE.FILE

Use ANALYZE.FILE to return information about static or dynamic hashed files.

Syntax

ANALYZE.FILE [DICT] [*filename* [*filename*] ...] [STATISTICS | STATS]
[NO.PAGE] [LPTR [*n*]]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
<i>filename</i>	The name of the file to be analyzed. ANALYZE.FILE can also use an active select list, in which case you need not specify <i>filename</i> .
NO.PAGE	Suppresses automatic paging.
LPTR	Sends output to the printer via logical print channel <i>n</i> .
STATISTICS	Lists additional information about dynamic files. See “Description.”
STATS	Same as STATISTICS.

ANALYZE.FILE Parameters

Description

For each dynamic file specified, a report lists the hashing algorithm, the modulo, the minimum modulus, the large record size, the group size, the split load, the merge load, the current load, the number of secondary indexes, and the size of the file. In NLS mode, the report includes the name of the file’s map.

For static hashed files, the ANALYZE.FILE report is the same as the FILE.STAT report.

Using the STATISTICS Keyword

When you use the STATISTICS keyword, the report takes more time to compile. The report is listed on the screen in four pages. Press ENTER to display the pages. See the second example for details of this report.

Examples

This example analyzes the dynamic file ORDERS:

```
>ANALYZE.FILE ORDERS
File name ..... ORDERS
Pathname ..... ORDERS
File type ..... DYNAMIC
Hashing Algorithm ..... GENERAL
No. of groups (modulus) .... 16 current ( minimum 1 )
Large record size ..... 1628 bytes
Group size ..... 2048 bytes
Load factors ..... 80% (split), 50% (merge) and 76%
(actual)
Total size ..... 43008 bytes
>
```

The second example analyzes the same ORDERS file, using the keyword STATISTICS. The first page of the statistics report displays the progress of the data compilation. After the data is compiled, press ENTER to continue.

```
File name ..... = ORDERS
File has 16 groups (each * represents 10 groups analyzed).
*
```

The second page of the statistics report includes information from the standard report as well as the total number of records, the number of large records, the number of deleted records, the total size of record data and record IDs, the unused space, and the total space for records:

```
File name ..... ORDERS
Pathname ..... ORDERS
File type ..... DYNAMIC
Hashing Algorithm ..... GENERAL
No. of groups (modulus) .... 16 current ( minimum 1, 0 empty,
                                1 overflowed, 0 badly
)
Number of records ..... 663
Large record size ..... 1628 bytes
Number of large records .... 0
Group size ..... 2048 bytes
Load factors ..... 80% (split), 50% (merge) and 76%
```

```
(actual)
Total size ..... 43008 bytes
Total size of record data .. 20359 bytes
Total size of record IDs ... 4553 bytes
Unused space ..... 14000 bytes
Total space for records .... 38912 bytes
```

The third page of the statistics report displays information about records and groups. For all groups, the report lists the average, minimum, maximum, and standard deviation of the number of disk records, records, large records, deleted records, data bytes, record ID bytes, unused bytes, and total bytes. For all records, the report lists the average, minimum, maximum, and standard deviation of the number of data bytes, record ID bytes, unused bytes, and total bytes.

File name	ORDERS			
	Number per group (total of 16 groups)			
	Average	Minimum	Maximum	StdDev
Group buffers	1.06	1	2	0.24
Records	41.44	27	53	6.60
Large records	0.00	0	0	0.00
Data bytes	1272.44	760	1901	285.18
Record ID bytes	284.56	216	411	48.46
Unused bytes	619.00	116	1784	401.19
Total bytes	2176.00	2048	4096	0.00
	Number per record (total of 663 records)			
	Average	Minimum	Maximum	StdDev
Data bytes	30.71	19	196	19.30
Record ID bytes	6.87	1	20	3.66
Total bytes	37.57	20	216	20.45

The last page of the report displays a histogram of the size of each record:

```
File name ..... ORDERS  
Histogram of record and ID lengths
```

58.2%

```
Bytes -----  
  
    up to 4|  
    up to 8|  
   up to 16|  
  up to 32| >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>  
  up to 64| >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>  
up to 128| >>>>>>>  
up to 256| >  
up to 512|  
   up to 1K|  
   up to 2K|  
   up to 4K|  
   up to 8K|  
  up to 16K|  
      More|
```

analyze.shm

Use *analyze.shm* as a diagnostic tool to examine information in the disk and the printer shared memory segments. Use *analyze.shm* from an operating system command prompt.

Syntax

analyze.shm { *options* }

Parameters

You must specify at least one option and precede each option with a hyphen. *options* are as follows:

Parameter	Description
-a [<i>seg#</i>]	(All) Lists information generated by all options. <i>seg#</i> specifies a printer memory segment. You must be UniVerse Administrator to use the <i>seg#</i> argument.
-b	(BASIC) Lists the status of all cataloged programs currently loaded in BASIC shared memory.
-c	(Configuration) Lists currently active authorization parameters.
-d	(Dynamic) Lists the status of all active dynamic file control blocks.
-f	(File) Lists all active file locks. The UniVerse LIST.READU command also lists this information.
-g	(Group) Lists all active group locks. The UniVerse LIST.READU command also lists this information.
-l	(Logging) Lists information about the transaction logging subsystem.
-L	Lists all NLS locales in shared memory.
-M	Lists all NLS character set maps in shared memory.

analyze.shm Parameters

Parameter	Description
<code>-n</code>	(Numbers) Lists raw, unformatted data for all table entries, including unused entries.
<code>-p [seg#]</code>	(Printer) Lists information about your printer memory segment. A UniVerse Administrator can use the <code>-p</code> option to list information about other users' printer memory segments. <i>seg#</i> specifies a printer memory segment.
<code>-r</code>	(Readu) Lists all active record locks. The UniVerse LIST.READU command also lists this information.
<code>-R</code>	(Replication) Lists information about the state of replication.
<code>-s</code>	(Semaphore) Lists the status of system semaphores. The UniVerse SEMAPHORE.STATUS command also lists this information.
<code>-t [0]</code>	(Tunables) Lists current configurable parameter values. 0 suppresses display of the asterisk that indicates the parameter has been changed from the default.
<code>-u</code>	(User) Lists the status of task synchronization locks. The UniVerse LIST.LOCKS command also lists this information.
<code>-x</code>	Lists general system information not displayed by the other options.
<code>-z</code>	Includes network licensed users, when used with <code>-c</code> and <code>-x</code> .

analyze.shm Parameters (Continued)

Description

The disk shared memory segment is a global work area that must be present for UniVerse to work. The printer shared memory segment is local to each user.

Use the [ANALYZE.SHM](#) command in the UniVerse environment to examine the disk and printer shared memory segments.

Examples

See the [CONFIG](#) command for an example of the report produced by *analyze.shm -c*.

See the [LIST.LOCKS](#) command for an example of the report produced by *analyze.shm -u*.

See the [LIST.READU](#) command for an example of the report produced by the `-f`, `-g`, and `-r` options of *analyze.shm*.

See the [SEMAPHORE.STATUS](#) command for an example of the report produced by *analyze.shm -s*.

All the examples show output from a UNIX system.

Catalog Shared Memory.

The `-b` option shows the contents of catalog shared memory. This is useful when you want to see what programs reside in catalog shared memory, which programs are in use, and which are available.

```
$ analyze.shm -b

State of shared memory: 4
Number of programs loaded into shared memory: 3
Size      References  Users   Pathname
  933             0       0
/usr/ardent/uv/catdir/*gtar*GTAR.858/7.1
  895             0       0
/usr/ardent/uv/catdir/*gtar*GTAR.858/7.2
 1055             0       0   /u1/gtar/PGMS.O/GTAR.9663.2
```

Printer Shared Memory

The `-p` option shows a user's printer memory segment followed by the user's segment's ID in decimal. The hexadecimal format in the output indicates the key of the segment.

If you specify the segment ID after the *analyze.shm -p* command, the ID is the shared memory key of that segment. If you do not specify the segment ID, the segment of the shared memory analysis tool is used and you see the default printer settings.

The following command displays a memory map showing used and unused sections for a user's printer shared memory segment. The terminal driver (BSD or SYSV) is included.

```
$ analyze.shm -p
Printer shared memory segment:
shared memory segment address=0x4475000
uid of owner=702 walter
gid of owner=700 doc
uid of creator=702 walter
gid of creator=700 doc
access mode (in octal)=0600
```

```
sequence number=0
shared memory segment id=428
shared memory key=0xACEB1E4C
size of printer segment in bytes=5120
pid of last shmop (this smat tool)=2508
pid of creator=2508
number of current attaches=1
last attach time=Tue Jul 15 17:00:25 1997
last deattach time=Wed Dec 31 19:00:00 1969
last change time=Tue Jul 15 17:00:25 1997
Pdata settings:
This machine uses a SYSV type terminal driver
fork_flag: 0
data_flag: 0
data_file: ""
term.crt.width: 80
term.crt.depth: 24
term.crt.top_mar: 0
term.crt.bot_mar: 0
term.lptr.width: 132
term.lptr.depth: 66
term.lptr.bot_mar: 3
term.flag: 0
term.Pinchr: 0 0 0 0
como_name: ""
break_disable: 0
date_format: 0
precision: 4
echo_chan: -1
wait_mode: 1
tand_flag: 0
tand_tty: ""
cont_tty: "/dev/pts/78"
src_set: 0
src_value: 0
err_value: 0
paging: 1
emulated_tty.mode.type: 3
emulated_tty.mode.min: 0
emulated_tty.mode.time: 0
emulated_tty.cc[CC_INTR]: 3
emulated_tty.cc[CC_QUIT]: 31
emulated_tty.cc[CC_SUSP]: -1
emulated_tty.cc[CC_DSUSP]: -1
emulated_tty.cc[CC_SWITCH]: 0
emulated_tty.cc[CC_ERASE]: 8
emulated_tty.cc[CC_WERASE]: -1
emulated_tty.cc[CC_KILL]: 21
emulated_tty.cc[CC_LNEXT]: -1
emulated_tty.cc[CC_REPRINT]: -1
emulated_tty.cc[CC_EOF]: 4
emulated_tty.cc[CC_EOL]: 0
emulated_tty.cc[CC_EOL2]: 0
emulated_tty.cc[CC_FLUSH]: -1
```



```

emulated_tty.cc[CC_START]: 17
emulated_tty.cc[CC_STOP]: 19
emulated_tty.cc[CC_LCONT]: 31
emulated_tty.cc[CC_FMC]: 30
emulated_tty.cc[CC_VMC]: 29
emulated_tty.cc[CC_SMC]: 28
emulated_tty.cc[CC_TMC]: 20
emulated_tty.cc[CC_SQLNULL]: 14
emulated_tty.cc[CC_CCDEL]: 0
emulated_tty.protocol.line: 1
emulated_tty.protocol.baud: 13
emulated_tty.protocol.data: 8
emulated_tty.protocol.stop: 1
emulated_tty.protocol.outenp: FALSE
emulated_tty.protocol.outodd: FALSE
emulated_tty.protocol.inipck: FALSE
emulated_tty.protocol.inmark: FALSE
emulated_tty.protocol.inigp: FALSE
emulated_tty.protocol.strip: FALSE
emulated_tty.output.post: TRUE
emulated_tty.output.tilde: FALSE
emulated_tty.output.bg: FALSE
emulated_tty.output.cs: TRUE
emulated_tty.output.tab: TRUE
emulated_tty.carrier.local: TRUE
emulated_tty.carrier.receive: TRUE
emulated_tty.carrier.hangup: TRUE
emulated_tty.crmode.inlcr: FALSE
emulated_tty.crmode.igncr: FALSE
emulated_tty.crmode.icrnl: TRUE
emulated_tty.crmode.onlcr: 1
emulated_tty.crmode.onocr: FALSE
emulated_tty.crmode.onlret: FALSE
emulated_tty.delay.bs: FALSE
emulated_tty.delay.cr: 0
emulated_tty.delay.ff: FALSE
emulated_tty.delay.lf: 0
emulated_tty.delay.vt: FALSE
emulated_tty.delay.tab: 0
emulated_tty.delay.fill: 0
emulated_tty.echo.on: TRUE
emulated_tty.echo.erase: TRUE
emulated_tty.echo.kill: 1
emulated_tty.echo.ctrl: TRUE
emulated_tty.echo.lf: FALSE
emulated_tty.handshake.xon: TRUE
emulated_tty.handshake.startany: FALSE
emulated_tty.handshake.tandem: TRUE
emulated_tty.handshake.dtr: FALSE
emulated_tty.signals.enable: TRUE
emulated_tty.signals.flush: TRUE
emulated_tty.signals.brkkey: 2
emulated_tty.ucase.ucin: FALSE
emulated_tty.ucase.ucout: FALSE

```

```
emulated_tty.ucase.xcase: FALSE
emulated_tty.ucase.invert: TRUE
physical_tty.mode.type: 0
physical_tty.mode.min: 0
physical_tty.mode.time: 0
physical_tty.cc[CC_INTR]: 3
physical_tty.cc[CC_QUIT]: 31
physical_tty.cc[CC_SUSP]: -1
physical_tty.cc[CC_DSUSP]: -1
physical_tty.cc[CC_SWITCH]: 0
physical_tty.cc[CC_ERASE]: 8
physical_tty.cc[CC_WERASE]: -1
physical_tty.cc[CC_KILL]: 21
physical_tty.cc[CC_LNEXT]: -1
physical_tty.cc[CC_REPRINT]: -1
physical_tty.cc[CC_EOF]: 4
physical_tty.cc[CC_EOL]: 0
physical_tty.cc[CC_EOL2]: 0
physical_tty.cc[CC_FLUSH]: -1
physical_tty.cc[CC_START]: 17
physical_tty.cc[CC_STOP]: 19
physical_tty.cc[CC_LCONT]: -1
physical_tty.cc[CC_FMC]: -1
physical_tty.cc[CC_VMC]: -1
physical_tty.cc[CC_SMC]: -1
physical_tty.cc[CC_TMC]: -1
physical_tty.cc[CC_SQLNULL]: -1
physical_tty.cc[CC_CCDEL]: -1
physical_tty.protocol.line: 1
physical_tty.protocol.baud: 13
physical_tty.protocol.data: 8
physical_tty.protocol.stop: 1
physical_tty.protocol.outenp: FALSE
physical_tty.protocol.outodd: FALSE
physical_tty.protocol.inipck: FALSE
physical_tty.protocol.inmark: FALSE
physical_tty.protocol.inigp: FALSE
physical_tty.protocol.strip: FALSE
physical_tty.output.post: TRUE
physical_tty.output.tilde: FALSE
physical_tty.output.bg: FALSE
physical_tty.output.cs: FALSE
physical_tty.output.tab: TRUE
physical_tty.carrier.local: TRUE
physical_tty.carrier.receive: TRUE
physical_tty.carrier.hangup: TRUE
physical_tty.crmode.inlcr: FALSE
physical_tty.crmode.igncr: FALSE
physical_tty.crmode.icrnl: TRUE
physical_tty.crmode.onlcr: 1
physical_tty.crmode.onocr: FALSE
physical_tty.crmode.onlret: FALSE
physical_tty.delay.bs: FALSE
physical_tty.delay.cr: 0
```

```

physical_tty.delay.ff: FALSE
physical_tty.delay.lf: 0
physical_tty.delay.vt: FALSE
physical_tty.delay.tab: 0
physical_tty.delay.fill: 0
physical_tty.echo.on: TRUE
physical_tty.echo.erase: TRUE
physical_tty.echo.kill: 1
physical_tty.echo.ctrl: FALSE
physical_tty.echo.lf: FALSE
physical_tty.handshake.xon: TRUE
physical_tty.handshake.startany: FALSE
physical_tty.handshake.tandem: TRUE
physical_tty.handshake.dtr: FALSE
physical_tty.signals.enable: TRUE
physical_tty.signals.flush: TRUE
physical_tty.signals.brkkey: 2
physical_tty.ucase.ucin: FALSE
physical_tty.ucase.ucout: FALSE
physical_tty.ucase.xcase: FALSE
physical_tty.ucase.invert: FALSE
Default transaction isolation level: 0
NLS per user override off: FALSE
Pblock for default printer channel:
inuse: FALSE
active: FALSE
hold: FALSE
nobuffer: FALSE
started: FALSE
format: TRUE
keepform: TRUE
primed: FALSE
wait: FALSE
spool: TRUE
hp: FALSE
ihold: FALSE
nohead: FALSE
eject: TRUE
inform: FALSE
ftn: FALSE
lnum: FALSE
next: FALSE
left: FALSE
whitespace: FALSE
chan: 0
copies: 0
spg: 0
epg: 0
width: 132
depth: 66
top_mar: 3
bot_mar: 3
lc: 0
flc: 0

```

```

hlc: 0
lpp: 0
mode: 1
priority: 0
level: 0
defer: 0
page: 0
sp_job_id: 0
fchan: 0
dchan: 0
head: ""
foot: ""
filename: ""
banner: ""
form: ""
lptr: ""
cr:
ff:
0x4475000                               Size Beginning of Memory Map
0x4475000-0x447500C 12 Used
0x447500C-0x4475018 12 Used
0x4475018-0x4475044 44 Used
0x4475044-0x4475070 44 Used
0x4475070-0x447509C 44 Used
0x447509C-0x4475A4C 2480 Free
0x4475A4C                               End of Memory Map

```

Dynamic File Table

The `-d` option shows the active dynamic file control blocks:

```

$ analyze.shm -d

Dynamic Files:
Slot # Inode Device Ref Count Splitload Mergeload Curmod Basemod Largerec
Filesp Nextsplit
    0 4232      5          1          19          80          50          357          256
590748      102
    1 26964     116          2          19          80          50          150          128
247476      23

```

Transaction Logging System

The `-l` option shows information about the transaction logging subsystem:

```

$ analyze.shm -l

Logging System State: 1
Logging System, Archiving: 0 Disk: 1 Tape: 0
Logging System, Checkpointing: 0
Logging system, last checkpointed log: 0
Logging system, active log: 0
Logging system, user request: 0

```

```

file flush: 0
checkpoint daemon fatal: 0
last log used: 0
large record logging: 0
log writes synchronous: 0
log daemon pid: 0
checkpoint daemon pid: 0
rollforward daemon pid: 0
last log synced: 0
next log to checkpoint: 0
lowest transaction id: 687
stale transaction: 687
semaphore id: -1
buffer size: 4096
block size: 512
previous bytes put: 0
bytes put in log: 0
bytes written in log: 0
current log file size: 0

```

General System Information. The `-x` option shows general system information not shown by other options:

```
$ analyze.shm -x
```

```

General System Information:
Login count: 6
Package "-UVNET" login count: 0
Package "-GCI" login count: 0
Package "-NLS" login count: 0
Package "-UCI" login count: 0
Package "-UVCS" login count: 0
Package "-UVADM" login count: 0
Base address for printer segment: 0x4475000
UniVerse home directory: /curbuild/uvnls/uv/
Shared Catalog: 0
Logging System State: 1
Logging System, Archiving: 0 Disk: 43 Tape: 10
Logging System, Checkpointing: 0
Logging system, last checkpointed log: 0
Logging system, active log: 0
Logging system, user request: 0
Next available transaction ID: 688
NT service process id: 0
NT service remote thread routine address: 0
Deadlock Daemon pid: 0
Replication system, logstate: 0
Replication system, repstate: 0
Replication system, log daemon pid: 0
Replication system, rep daemon pid: 0
Spare7: 0
Spare8: 0
Spare9: 0
Spare10: 0
Spare11: 0
Spare12: 0
Spare13: 0
Spare14: 0
Semaphore debugging: 0
Uvnetsd debugging: 0

```

```
Uvnetlicd debugging: 0
Uvnetlicd watchdog: 0
Uvserver debugging: 0
Feature6: 0
Feature7: 0
Feature8: 0
Feature9: 0
Feature10: 0
Feature11: 0
Feature12: 0
Feature13: 0
Feature14: 0
Feature15: 0
Feature16: 0
The next example shows information about the state of replication:
Replication mode: 1
  logstate: 0
  repstate: 0
  logcontrol: 0, 0
  repcontrol: 0, 0
  log daemon pid: 0
  rep daemon pid: 0
  buffer size: 40960
  block size: 4096
  read offset: 0
  write offset: 0
```

ANALYZE.SHM

Use ANALYZE.SHM as a diagnostic tool to examine information in the disk and the printer shared memory segments.

Syntax

ANALYZE.SHM { *options* }

Parameters

You must specify at least one option. Specify options in lowercase and precede them with hyphens. The following table describes each parameter of the syntax.

Parameter	Description
-a [<i>seg#</i>]	(All) Lists information generated by all options. <i>seg#</i> specifies a printer memory segment. You must be a UniVerse Administrator to use the <i>seg#</i> argument.
-b	(BASIC) Lists the status of all cataloged programs currently loaded in BASIC shared memory.
-c	(Configuration) Lists currently active authorization parameters.
-d	(Dynamic) Lists the status of all active dynamic file control blocks.
-f	(File) Lists all active file locks. The LIST.READU command also lists this information.
-g	(Group) Lists all active group locks. The LIST.READU command also lists this information.
-l	(Logging) Lists information about the transaction logging subsystem.
-L	Lists all NLS locales in shared memory.
-M	Lists all NLS character set maps in shared memory.
-n	(Numbers) Lists raw, unformatted data for all table entries, including unused entries.

ANALYZE.SHM Parameters

Parameter	Description
<code>-p [seg#]</code>	(Printer) Lists information about your printer memory segment. A UniVerse Administrator can use the <code>-p</code> option to list information about other users' printer memory segments. <i>seg#</i> specifies a printer memory segment.
<code>-r</code>	(Readu) Lists all active record locks. The <code>LIST.READU</code> command also lists this information.
<code>-R</code>	(Replication) Lists information about the state of replication.
<code>-s</code>	(Semaphore) Lists the status of system semaphores. The SEMAPHORE.STATUS command also lists this information.
<code>-t [0]</code>	(Tunables) Lists current configurable parameter values. 0 suppresses display of the asterisk that indicates the parameter has been changed from the default.
<code>-u</code>	(User) Lists the status of task synchronization locks. The LIST.LOCKS command also lists this information.
<code>-x</code>	Lists general system information not displayed by the other options.
<code>-z</code>	Includes network licensed users, when used with <code>-c</code> and <code>-x</code> .

ANALYZE.SHM Parameters (Continued)

Description

The disk shared memory segment is a global work area that must be present for UniVerse to work. The printer shared memory segment is local to each user.

Use the *analyze.shm* command from an operating system command prompt to examine the disk and printer shared memory segments.

Examples

See the [CONFIG](#) command for an example of the report produced by `ANALYZE.SHM -c`.

See the [LIST.LOCKS](#) command for an example of the report produced by `ANALYZE.SHM -u`.

See the [LIST.READU](#) command for an example of the report produced by the `-f`, `-g`, and `-r` options of ANALYZE.SHM.

See the [SEMAPHORE.STATUS](#) command for an example of the report produced by ANALYZE.SHM `-s`.

All the examples show output from a UNIX system.

This example shows the contents of catalog shared memory:

```
>ANALYZE.SHM -b

State of shared memory: 4
Number of programs loaded into shared memory: 3
Size      References  Users  Pathname
 933              0      0
/usr/ardent/uv/catdir/*gtar*GTAR.858/7.1
 895              0      0
/usr/ardent/uv/catdir/*gtar*GTAR.858/7.2
1055              0      0  /ul/gtar/PGMS.O/GTAR.9663.2
```

The next example shows the active dynamic file control blocks:

```
>ANALYZE.SHM -d

Dynamic Files:
Slot # Inode Device Ref Count Splitload Mergeload Curmod Basemod
Largerec
Filesp Nextsplit
  0 4232      5          1          19          80          50          357
256
590748      102
  1 26964     116          2          19          80          50          150
128
247476      23
```

The next example shows information about the transaction logging subsystem:

```
>ANALYZE.SHM -l

Logging System State: 1
Logging System, Archiving: 0 Disk: 1 Tape: 0
Logging System, Checkpointing: 0
Logging system, last checkpointed log: 0
Logging system, active log: 0
Logging system, user request: 0
file flush: 0
checkpoint daemon fatal: 0
last log used: 0
large record logging: 0
log writes synchronous: 0
log daemon pid: 0
checkpoint daemon pid: 0
```

```

rollforward daemon pid: 0
last log synced: 0
next log to checkpoint: 0
lowest transaction id: 687
stale transaction: 687
semaphore id: -1
buffer size: 4096
block size: 512
previous bytes put: 0
bytes put in log: 0
bytes written in log: 0
current log file size: 0

```

The next example shows general system information not shown by other options:

>ANALYZE.SHM -x

```

General System Information:
Login count: 6
Package "-UVNET" login count: 0
Package "-GCI" login count: 0
Package "-NLS" login count: 0
Package "-UCI" login count: 0
Package "-UVCS" login count: 0
Package "-UVADM" login count: 0
Base address for printer segment: 0x4475000
UniVerse home directory: /curbuild/uvnls/uv/
Shared Catalog: 0
Logging System State: 1
Logging System, Archiving: 0 Disk: 43 Tape: 10
Logging System, Checkpointing: 0
Logging system, last checkpointed log: 0
Logging system, active log: 0
Logging system, user request: 0
Next available transaction ID: 688
NT service process id: 0
NT service remote thread routine address: 0
Deadlock Daemon pid: 0
Replication system, logstate: 0
Replication system, repstate: 0
Replication system, log daemon pid: 0
Replication system, rep daemon pid: 0
Spare7: 0
Spare8: 0
Spare9: 0
Spare10: 0
Spare11: 0
Spare12: 0
Spare13: 0
Spare14: 0
Semaphore debugging: 0
Uvnetd debugging: 0
Uvnetlicd debugging: 0
Uvnetlicd watchdog: 0

```

```
Uvserver debugging: 0
Feature6: 0
Feature7: 0
Feature8: 0
Feature9: 0
Feature10: 0
Feature11: 0
Feature12: 0
Feature13: 0
Feature14: 0
Feature15: 0
Feature16: 0
```

The next example shows information about the state of replication:

```
Replication mode: 1
  logstate: 0
  repstate: 0
  logcontrol: 0, 0
  repcontrol: 0, 0
  log daemon pid: 0
  rep daemon pid: 0
  buffer size: 40960
  block size: 4096
  read offset: 0
  write offset: 0
```

ASSIGN

Use ASSIGN to request exclusive control of a device.

Syntax

ASSIGN *device* TO MTU *n* [MAP *mapname*] [-WAIT] [BLK *size*]

ASSIGN *device* TO LPTR *n* [-WAIT]

Parameters

The following table describes each parameter of the syntax:

Parameter	Description
<i>device</i>	Specifies the device you want to assign. <i>device</i> must be the record ID of a device definition in the &DEVICE& file. The device can be a magnetic tape unit or a line printer. On UNIX systems, the device can also be a file, a serial port, or any other device defined in the &DEVICE& file.
MTU <i>n</i>	Specifies the logical tape unit. <i>n</i> can be a number from 0 through 7. The number of the logical tape unit is used in commands like T.REW or a BASIC statement like READT statement to access the assigned device.
LPTR <i>n</i>	Specifies a logical print channel. <i>n</i> can be a number from 0 through 255. The logical print channel number is used in Retrieve commands or in BASIC PRINT statement ON to access the assigned device.
MAP <i>mapname</i>	Specifies that you want to set a map for the device. <i>mapname</i> must be built and installed in shared memory. <i>mapname</i> can also be one of the following:
NONE	Specifies the UniVerse internal character set.
UTF8	Specifies the UniVerse internal character set but with system delimiters mapped to the Unicode Private Use Area.

ASSIGN Parameters

Parameter	Description
	DEFAULT Specifies <i>mapname</i> as the name in the NLSDEFDEVMAP parameter in the <i>uvconfig</i> file. –WAIT Tells the processor to queue this assignment if the device is currently assigned to another user. After the other user unassigns the device, it is assigned to you. BLK <i>size</i> Specifies the block size of the physical tape record, in bytes. If you do not specify block size, the default block size defined in the &DEVICE& file is used.

ASSIGN Parameters (Continued)

Description

If you do not use the –WAIT option and the device you requested is assigned to another user, a message like the following appears:

```
Device 'device' is currently ASSIGNED to: User Number nn
```

In NLS mode, the map name specified by ASSIGN overrides any map name specified in the &DEVICE& file record for *device* until the device is unassigned or another ASSIGN command specified another map.

If you do not assign a device before using any I/O operation that requires exclusive use of a device, a message like the following appears:

```
Magnetic tape rewind failed. No device ASSIGNED to MTU
```

Use the ASSIGN command, then repeat the operation that you attempted.

The device must be set with the proper permissions for you to assign it.

Use the [UNASSIGN](#) command to release control of a device after using the ASSIGN command.

Examples

To get control of tape drive MT1, enter the following:

```
>ASSIGN MT1 TO MTU 0 BLK 8192
```

To output to a serial printer on a serial port, enter the following:

```
>ASSIGN TTY04 TO LPTR 0  
>LIST VOC LPTR
```

The following example gets control of tape drive MT2, setting *mapname* to the ISO8859-1 map:

```
>ASSIGN MT2 TO MTU 1 MAP ISO8859-1
```

AUTOLOGOUT

Use AUTOLOGOUT to enable or disable automatic logout. With AUTOLOGOUT enabled, UniVerse logs you out automatically if you have not pressed a key within a specified time.

Syntax

AUTOLOGOUT [*time*]

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>time</i>	The number of minutes that must elapse before automatic logout occurs. If you specify 0, automatic logout is disabled. If you do not specify <i>time</i> , the current status of automatic logout is displayed.

AUTOLOGOUT Parameter

Examples

```
>AUTOLOGOUT
Automatic logout is disabled.
>AUTOLOGOUT 40
Automatic logout is set for 40 minutes.
```

AVAIL

Use AVAIL to display statistics about available disk space, including the number of bytes used, the number of bytes still available, and the percent of total disk space used.

Syntax

AVAIL [*device*]

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>device</i>	UNIX. The file system mounted on a system. Windows Platforms. The drive letter or UNC drive name on a system. If <i>device</i> is not specified for a Windows system, the output contains statistics for each connected drive.

AVAIL Parameter

Examples

UNIX. The AVAIL command executes the *df* command, which produces a report similar to this:

>AVAIL

Filesystem	kbytes	used	avail	capacity	Mounted on
/dev/ioc/cdisk00a	9102	14080	3110	82%	/
/dev/ioc/cdisk00c	7950	41528	1626	96%	/usr
/dev/ioc/cdisk00g	89094	238894	21290	92%	/cs
/dev/ioc/cdisk01a	9158	15172	2070	88%	/mktg
/dev/ioc/cdisk01c	8070	39426	3836	91%	/qa
/dev/ioc/cdisk01g	89094	247378	12806	95%	/rd

AVAIL and *df* produce this report in different formats depending on the UNIX system.

Windows Platforms. The AVAIL command on a system produces a report similar to this:

>AVAIL

Drive	Type	Total KBytes	Used KBytes	%Used
a:\	floppy	No information		
c:\	local	104871	92284	87%
d:\	local	946110	831105	87%
e:\	remote	1083162	1042888	96%
h:\	remote	1051705	795814	75%

BASIC

Use BASIC to compile a BASIC program.

Syntax

BASIC *filename* [*programs* | *] [*options*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>filename</i>	The name of a type 1 or type 19 file containing BASIC programs. Filenames can include only ASCII characters, no multibyte characters.
<i>programs</i>	The record IDs of the programs to be compiled. Program names can include only ASCII characters, no multibyte characters. You can compile more than one program at a time, but the programs must all be in the same file. If a select list is active, you need not specify <i>programs</i> .
*	Specifies all programs in the file.

BASIC Parameters

The following table describes available options.

Option	Description
<code>+\$option</code>	Turns on the specified \$OPTIONS option. <i>option</i> can also be the name of a UniVerse flavor. See the \$OPTIONS statement for the list of options and UniVerse flavors. You must specify all options you want to turn on before the options you want to turn off.
<code>-\$option</code>	Turns off the specified \$OPTIONS option. <i>option</i> can also be the name of a UniVerse flavor. See the \$OPTIONS statement for the list of options and UniVerse flavors. You must specify all options you want to turn off after the options you want to turn on.
<code>-I</code>	Suppresses execution of RAID or VLIST on a compiler or a BASIC program.
<code>-L</code>	Generates a listing of the program.
<code>-LIST</code>	Same as <code>-L</code> .
<code>-X</code>	Generates a cross-reference table of statement labels and variable names used in the program.
<code>-XREF</code>	Same as <code>-X</code> .
<code>-S</code>	Generates a listing of the program and spools it directly to the printer rather than to a file.
<code>-SPOOL</code>	Same as <code>-S</code> .
<code>-T</code>	Suppresses the symbol and line number tables that are usually appended to the end of an object file for run-time error messages.

BASIC Options

Description

The object code produced by the compiler is saved in a file with the source *filename* and the suffix `.O` (for example, `BP.O`). The record ID of the object code is the same as the record ID of the source code.

A listing produced with either the `-LIST` or the `-XREF` option is saved in a file with the source *filename* and a suffixed `.L` (for example, `BP.L`). The record ID in the listing file is the same as the record ID of the source code.

Example

This example compiles PROG1 and spools a listing of PROG1 to the printer:

```
>BASIC BP PROG1 -SPOOL
```

BELL

Use BELL to specify whether the terminal bell sounds when UniVerse generates warning messages.

Syntax

BELL { ON | OFF }

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
ON	(Default) Specifies that the terminal bell sounds.
OFF	Specifies that the terminal bell does not sound.

BELL Parameters

BLOCK.PRINT

Use BLOCK.PRINT to print block characters on the printer.

Syntax

BLOCK.PRINT *string* [*string* ...]

Parameters

The following table the parameter of the syntax.

Parameter	Description
<i>string</i>	An unquoted ASCII 7-bit character string with no embedded blanks. You can print several character strings on separate lines by specifying them on the command line separated by blanks.

BLOCK.PRINT Parameter

Description

To use BLOCK.PRINT, your account must have a VOC file entry pointing to the BLTRS file in the UV account. BLTRS contains each character that can be listed in block format. If the character is not in BLTRS, BLOCK.PRINT cannot print it. To list the characters you can print in block format, enter **SORT BLTRS** at the system prompt.

The characters that you include in the command line are printed on the printer in a block format. Each character of the string is nine lines long and between five and twenty characters wide. The width varies according to the relative width of the letter that is printed. The characters are printed as uppercase letters. The widest letters are W and M; the narrowest letter is I.

Because character width varies, the total length of the character string that is printed also varies. The length of the string depends on the letters within the string and cannot be longer than the printer's line width. If the string is too long, BLOCK.PRINT displays an error message.

To display block characters on the terminal, use the [BLOCK.TERM](#) command.

Example

```
>BLOCK.PRINT ABC
```

AAAA	BBBBBBBBBB	CCCCCCCC
AAAAAA	BBBBBBBBBB	CCCCCCCC
AAAAAAA	BBBB BBB	CCCC CCCC
AAAA AAA	BBBBBBBBBB	CCCC CCCC
AAAA AAA	BBBBBBBBBB	CCCC CCCC
AAAAAAAAA	BBBB BBB	CCCC CCCC
AAAAAAAAA	BBBB BBB	CCCC CCCC
AAAA AAA	BBBBBBBBBB	CCCCCCCC
AAAA AAA	BBBBBBBBBB	CCCCCCCC

BLOCK.TERM

Use BLOCK.TERM to display block characters on the terminal.

Syntax

BLOCK.TERM *string* [*string ...*]

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>string</i>	An unquoted ASCII 7-bit character string with no embedded blanks. You can display several character strings on separate lines by specifying them on the command line separated by blanks.

BLOCK.TERM Parameter

Description

To use BLOCK.TERM, your account must have a VOC file entry pointing to the BLTRS file in the UV account. BLTRS contains each character that can be listed in block format. If the character is not in BLTRS, BLOCK.TERM cannot display it. To list the characters you can display in block format, enter **SORT BLTRS** at the system prompt.

The characters that you include in the command line are displayed on the terminal in a block format. Each character in the string is nine lines long and between five and twenty characters wide. The width varies according to the relative width of the letter that is displayed. The characters are printed as uppercase letters. The widest letters are W and M; the narrowest letter is I.

Because character width varies, the total length of the character string that is displayed also varies. The length of the string depends on the letters within the string and cannot be longer than the terminal's line width. If the string is too long, BLOCK.TERM displays an error message.

To print block characters on the printer, use the [BLOCK.PRINT](#) command.

Example

```
>BLOCK . TERM ABC

      AAAA      BBBBBBBBBB      CCCCCCCCCC
      AAAAAA      BBBBBBBBBBBB      CCCCCCCCCCCC
      AAAAAAAAA      BBBB      BBBB      CCCC      CCCC
      AAAA      AAAA      BBBBBBBBBBBB      CCCC      CCCC
      AAAA      AAAA      BBBBBBBBBB      CCCC
      AAAAAAAAAAA      BBBB      BBBB      CCCC      CCCC
      AAAAAAAAAAA      BBBB      BBBB      CCCC      CCCC
      AAAA      AAAA      BBBBBBBBBBBB      CCCCCCCCCCCC
      AAAA      AAAA      BBBBBBBBBB      CCCCCCCCCC

>
```

BREAK

Use BREAK to enable or disable the Intr, Stop, Susp, and Break keys.

Syntax

BREAK { ON | OFF | COUNT }

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
ON	Enables the Intr, Stop, Susp, and Break keys.
OFF	Disables the Intr, Stop, Susp, and Break keys.
COUNT	Returns the number of BREAK OFF commands issued during the current session.

BREAK Parameters

Description

When these keys are disabled, typing one of them has no effect on the current process. The keys are enabled by default.

BUILD.INDEX

Use BUILD.INDEX to build a secondary index from a file. You must use this command to build a newly created index for a file that already contains records. To build an index on any data file, you must have write permissions on the file dictionary. To build an index on any file dictionary, you must have write permissions on the DICT.DICT file in the UV account.

Syntax

BUILD.INDEX [DICT] [*filename*] [*indexes* | ALL]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
<i>filename</i>	The name of a UniVerse file for which to build a secondary index. If <i>filename</i> is not specified, BUILD.INDEX prompts for it.
<i>indexes</i>	A list of fields in <i>filename</i> used as secondary index keys. Index names should be separated by spaces.
ALL	Specifies that all the secondary index keys in the file should be built.

BUILD.INDEX Parameters

Description

If you do not specify either *indexes* or ALL, you are prompted to enter the name of an index.

BUILD.INDEX first removes the contents of all specified indexes, then completely rebuilds the indexes.

Use BUILD.INDEX after you create a new secondary index to a file that already contains records. The file is locked for use by other users during the building process.

Example

```
>BUILD.INDEX ORDERS CUST.ID
```

```
Locking 'ORDERS' file for exclusive use.  
Starting SSELECT for file 'ORDERS index CUST.ID'.  
Compiling "@INDEX.CUST.ID".  
@Ak.0 ; splice ( @1 , ( char ( 251 ) ) , @ID )
```

```
956 record(s) selected to SELECT list #1.
```

```
Clearing Index File INDEX.000
```

```
Starting DATA processing for index 'CUST.ID'!
```

```
956 total processed.
```

```
Updating INDEX.MAP flags...
```

```
Index build of CUST.ID complete.
```

```
File 'ORDERS' Unlocked.
```

```
>
```

CATALOG

Use CATALOG to catalog a BASIC program. Cataloging a program makes it available to all users or to users of one account. You must catalog a program before another BASIC program can call it as an external subroutine.

Syntax

CATALOG [*filename*] [[*catalog.name*] *program.name* | *] [*options*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>filename</i>	The name of the program source code file containing the programs to be cataloged. File names can include only ASCII characters, no multibyte characters. If you do not specify <i>filename</i> , you cannot specify other qualifiers on the command line. In this case CATALOG prompts you for the qualifier values, one at a time.
<i>catalog.name</i>	The name the program will have in the catalog. Catalog names can include only ASCII characters, no multibyte characters. This name must be used in a CALL statement or as a verb to reference the cataloged program. If you do not specify <i>catalog.name</i> , <i>program.name</i> is the name the program will have in the catalog.
<i>program.name</i>	The name of the program in <i>filename</i> to be cataloged. Program names can include only ASCII characters, no multibyte characters. You can specify only one program name on the command line unless you enter an asterisk (*) to specify all programs in a file. CATALOG can also read from an active select list.
*	Specifies all programs in <i>filename</i> .

CATALOG Parameters

You can specify one or more of the following *options*:

Option	Description
FORCE	Specifies that CATALOG should overwrite any cataloged program with the same <i>catalog.name</i>. If you do not specify FORCE and a globally cataloged program with the same name exists, CATALOG asks for permission to overwrite it. When a normally cataloged program is cataloged, a program with the same <i>catalog.name</i> is overwritten. Locally cataloged programs are overwritten as soon as they are recompiled. You cannot specify FORCE at a prompt.
NOXREF	Specifies that the program should be cataloged without the cross-reference and symbol table information. This makes it difficult to use any of the UniVerse debugging tools and should be specified only after a program has been thoroughly tested. You cannot specify NOXREF at a prompt.
LOCAL	Specifies that the program is to be cataloged in the current account instead of in the system catalog space. You can specify the LOCAL keyword in response to the prompt for <i>catalog.name</i>.
COMPLETE	Specifies that the VOC entry for a locally cataloged program should be the absolute pathname. The VOC entry normally specifies the location of the program relative to the user's account.

CATALOG Options

Description

If you use CATALOG with no options, you are prompted to enter the file name, the catalog name, and the program name. If you press ENTER at any of the prompts, CATALOG terminates without cataloging anything.

You must compile the source code before you use the CATALOG command. UniVerse assumes that the object code to be cataloged is in the corresponding object code file named *filename.O*.

There are three ways to catalog a program: locally, normally (or standard), and globally. Each method has different implications.

Cataloging Locally

Local cataloging creates a VOC entry for the program. This entry is a verb that points to the file and record containing the object code for the cataloged program. You can access a locally cataloged program only from the account in which it was cataloged, unless you copy the VOC entry for the catalog name to another account. Because cataloging a program locally only creates a VOC entry pointing to the object file, you need not recatalog the program every time you recompile it.

To catalog a program locally, specify the LOCAL keyword on the command line or enter LOCAL at the following prompt:

```
Catalog name or LOCAL =
```

Cataloging Normally

Normal cataloging copies the specified object record to the system catalog space, making it available to all users. The name of the program in the catalog is in the following format:

```
*account*catalog.name
```

account is the name of the current account directory.

Normal cataloging also creates a VOC entry for the catalog name. This entry is a verb that contains the name **account*catalog.name* in field 2.

Because normal cataloging copies the object code to the system catalog space, you must recatalog the program every time you recompile it.

To catalog a program normally, specify a *catalog.name* that does not begin with the characters *, -, \$, or !, and do not specify the keyword LOCAL.

To avoid overwriting a cataloged program with another of the same name, you can use the IAM command to change the *account* name. For example, if you catalog the following two programs normally, they are given the catalog name **information*UPDATE*, and the most recently cataloged program overwrites the other:

```
/usr/accounts/information/BP/UPDATE  
/usr/salary/information/BP/UPDATE
```

Another way to avoid conflicting catalog names is to catalog the programs with different names. This example changes the account directory name before cataloging the UPDATE program from the BP file. The example assumes the user's account is in */usr/accounts/information*.

```
>WHO
56 information
>LOGTO /usr/salary/information
>IAM salary
>WHO
56 salary From information
>CATALOG BP UPDATE
"*salary*UPDATE" cataloged.
>
```

How you call or invoke a normally cataloged program depends on your location.

- From the account where it was cataloged: To call the program, use the BASIC CALL statement with *catalog.name*. To invoke the program from the system prompt, enter *catalog.name* at the prompt.
- From any other account: To call the program, use the CALL statement with the catalog name as listed in the catalog space (for example, **account*catalog.name*). To invoke the program from the system prompt, enter ***account*catalog.name** at the prompt.

You can also copy the VOC entry for the catalog name from the account where the program was cataloged to another account. If you do this, you can use *catalog.name* to call or invoke the program.

Cataloging Globally

Like normal cataloging, global cataloging copies the specified object record to the system catalog space, making it available to all users. The name of the program in the catalog is in the following format:

```
*catalog.name
-catalog.name
$catalog.name
!catalog.name
```

Global cataloging does not create a VOC entry for the catalog name. The UniVerse command processor and the run machine look in the system catalog space for verbs or external subroutines with names that have an initial *, -, \$, or ! character. Because globally cataloged subroutines are accessed without a VOC entry, they are available to all accounts on the system as soon as they are cataloged.

Because global cataloging copies the object code to the system catalog space, you must recatalog the program every time you recompile it.

To catalog a program globally, specify a *catalog.name* beginning with *, -, \$, or !, and do not specify the keyword LOCAL.

Examples

This example catalogs the UPDATE program normally, in the SALES account:

```
>CATALOG BP UPDATE
```

No catalog name is specified, so the program name is also the catalog name. It is listed in the catalog space as *SALES*UPDATE. From the SALES account it can be called using the name UPDATE. From any other account it can be called using the name *SALES*UPDATE.

The next example catalogs UPDATE globally as *UPDATE and automatically overwrites any existing *UPDATE without prompting:

```
>CATALOG BP *UPDATE UPDATE FORCE
```

The program is listed in the catalog as *UPDATE. Any user from any account can call this program using the name *UPDATE.

The next example catalogs UPDATE locally, setting up an entry in the VOC file called UPDATE:

```
>CATALOG BP UPDATE LOCAL  
"PROG3" cataloged.
```

This is the VOC entry:

```
      UPDATE  
0001  V  
0002  BP.O/UPDATE  
0003  B  
0004  BN  
0005  
0006  
0007  
0008  
0009  BP.O
```

This example shows a subroutine being cataloged:

```
>CATALOG  
Catalog name or LOCAL =*TEST  
File name           =BP  
Program name        =TEST  
"*TEST" already exists globally.  
Overwriting this file may affect other users.  
Do you want to overwrite the existing version? (Y/N) = N
```

CD

Use CD to compile the I-descriptors in a file dictionary before using them in a sentence. You can compile one, several, or all of the I-descriptors in the dictionary.

Syntax

CD *filename* [*descriptors*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>filename</i>	The name of the UniVerse file whose dictionary entries are to be compiled. Do not specify the DICT keyword.
<i>descriptors</i>	The names of the I-descriptors to be compiled. If you do not specify a descriptor name, all I-descriptors in the dictionary are compiled. You also can use an active select list to specify I-descriptors.

CD Parameters

Description

CD stores the compiled object code and the time and date of the compilation in the I-descriptor record. If you change the I-type expression, the Editor flags the I-descriptor and displays a reminder that the I-descriptor must be recompiled when you file the record. It invalidates the existing descriptor as well. Retrieve compiles this I-descriptor before using it in a sentence.

Because the I-descriptors in a dictionary are often related, compile them together.

CD is a synonym for the [COMPILE.DICT](#) command.

Example

This example compiles four I-descriptors:

```
>CD ORDERS
Compiling "*A9998".
@RECCOUNT
Compiling "EXT".
QTY * ( TRANS ( BOOKS , CODE , PRICE , X ) )
Compiling "PRICE".
TRANS ( BOOKS , CODE , PRICE , X )
Compiling "TITLE".
TRANS ( BOOKS , CODE , TITLE , X )
```

CENTURY.PIVOT

Use CENTURY.PIVOT to override the systemwide century pivot year defined in the *uvconfig* file.

Syntax

CENTURY.PIVOT [*year* | *nn*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>year</i>	A four-digit year. The first two digits specify the century, the last two digits specify the pivot year.
<i>nn</i>	A two-digit code indicating a sliding pivot year. <i>nn</i> can be 00 through 99.

CENTURY.PIVOT Parameters

Description

In UniVerse, when you enter as input a year in two-digit format (for example, 99 or 01), UniVerse by default assumes the following:

- Years entered in the range 30 through 99 stand for 1930 through 1999
- Years entered in the range 00 through 29 stand for 2000 through 2029

Administrators can change these default ranges in three ways:

- Setting or changing the CENTURYPIVOT configurable parameter in the *uvconfig* file (for information about configurable parameters, see *Administering UniVerse*).
- Using the CENTURY.PIVOT UniVerse command
- Using the CENTURY.PIVOT BASIC function (see *UniVerse BASIC*).

The CENTURYPIVOT configurable parameter sets the systemwide century pivot year for UniVerse. You can use the CENTURY.PIVOT command to override the century pivot year for the current session.

You can set the century pivot year in two ways:

Static Century Pivot Year

If you specify the century pivot year with four digits, the first two digits specify the century, and the last two digits specify the pivot year.

For example, if you specify *year* as 1940, two-digit years specified in the range of 40 through 99 stand for 1940 through 1999, and two-digit years specified in the range of 00 through 29 stand for 2000 through 2039. These ranges remain fixed until you explicitly change them.

Sliding Century Pivot Year

If you enter the century pivot year as a two-digit code (*nn*), the century pivot year changes relative to the current year. The formula for determining the century pivot year is as follows:

$$current.year - (100 - nn)$$

For example, if the current year is 2000 and *nn* is 05, the century pivot year is 1905. This means that two-digit years specified in the range of 05 through 99 stand for 1905 through 1999, and two-digit years specified in the range of 00 through 04 stand for 2000 through 2004.

If the current year is 2005 and *nn* is 05, the century pivot year is 1910. Two-digit years specified in the range of 10 through 99 stand for 1910 through 1999, and two-digit years specified in the range of 00 through 09 stand for 2000 through 2009.

If the current year is 2001 and *nn* is 30, the century pivot year is 1931. Two-digit years specified in the range of 31 through 99 stand for 1931 through 1999, and two-digit years specified in the range of 00 through 30 stand for 2000 through 2030.

CHAP

Use CHAP to control execution priority.

Syntax

CHAP [UP | DOWN]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
UP	Specifies that you want to improve your execution priority. You must be a UniVerse Administrator to specify CHAP UP.
DOWN	Specifies that you want to lower your execution priority. DOWN is the default.

CHAP Parameters



CHANGE.DOMAIN

Use CHANGE.DOMAIN to change a Windows domain name in the UV_USERS file of the SQL catalog. You must be a database administrator (DBA) to use CHANGE.DOMAIN.

Note: This command is not supported on UNIX systems.

Syntax

CHANGE.DOMAIN *domain new.domain*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>domain</i>	The domain name you want to change. It must match the name of the domain in the NAME column of the UV_USERS file.
<i>new.domain</i>	The new domain name.

CHANGE.DOMAIN

Description

On Windows platforms, the user name can include the domain name. For example, user *jdoe* might belong to the SALES domain, in which case the full user name could be SALES\jdoe. This user name is stored in the NAME column of the UV_USERS file in the SQL catalog.

Users sometimes change their domain. The CHANGE.DOMAIN command provides a simple way to change the domain entries in the UV_USERS file.

Example

The following SELECT statement extracts a user name from the UV_USERS file:

```
>SELECT NAME FROM UV_USERS;  
User Name.....  
  
SALES\jdoe
```

CHANGE.DOMAIN changes the domain name to MARKETING:

```
>CHANGE.DOMAIN SALES MARKETING  
UniVerse/SQL: 1 record updated.
```

SELECT now shows the new domain name:

```
>SELECT NAME FROM UV_USERS;  
User Name.....  
  
MARKETING\jdoe
```

CHDIR

Use CHDIR to switch from your current UniVerse account to another UniVerse account without exiting the current process.

Syntax

CHDIR *pathname*

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>pathname</i>	The path of the directory that contains the UniVerse account to which you want to switch.

CHDIR Parameter

Description

If the account is not set up for UniVerse, the following message appears:

```
Directory account is not set up for UniVerse
```

If the account you are moving to has a LOGIN entry in its VOC file, UniVerse executes it before displaying the > prompt.

Example

This example changes your working directory to */usr/acct/demo* without leaving UniVerse:

```
>CHDIR /usr/acct/demo
```

CHECK.SUM

Use CHECK.SUM to get statistical information on values in a particular field for one or more records in a file. This information includes the total number of bytes, the average number of bytes, the number of records analyzed, the checksum, and the bit count. The checksum is the result of multiplying the binary value of characters in a string by their positional value. The checksum has a high probability of being unique for a particular character string, making it useful for verifying data.

Syntax

```
CHECK.SUM [ DICT | USING [ DICT ] dictname ] filename
[ records | FROM n ] [ selection ] [ field ] [ report.qualifiers ]
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Checks records in the file dictionary of <i>filename</i> . If you do not specify DICT, records in the data file are checked.
USING [DICT] <i>dictname</i>	If DICT is not specified, uses <i>dictname</i> as the dictionary of <i>filename</i> . If DICT is specified, the dictionary of <i>dictname</i> is used as the dictionary of <i>filename</i> .
<i>filename</i>	The file whose records you want to checksum. You can specify <i>filename</i> anywhere in the sentence. CHECK.SUM uses the first word in the sentence that has a file descriptor in the VOC file as the file name.
<i>records</i>	Specifies the records to include. You can specify as many record IDs as you want, separated by spaces. Enclose record IDs in single quotation marks to ensure that Retrieve does not interpret them as field names or keywords.
FROM <i>n</i>	Specifies the records whose record IDs are stored in select list <i>n</i> .

CHECK.SUM Parameters

Parameter	Description
<i>selection</i>	A selection expression specifying the conditions that data in a record must meet for the record to be selected. A selection expression begins with the keyword WITH. For syntax details, see the WITH keyword.
<i>field</i>	The name of the field to be checksummed. You can specify only one field. If you do not specify a field name, CHECK.SUM uses the first field specified in the @ phrase. If there is no @ phrase in the file dictionary, CHECK.SUM uses the entire record.
<i>report.qualifiers</i>	One or more of the following keywords: FIRST LPTR ONLY SAMPLED ID.ONLY NOPAGE SAMPLE These keywords modify the report format.

CHECK.SUM Parameters (Continued)

Description

The CHECK.SUM command displays output in the following format:

```

BYTE STATISTICS FOR field | filename
TOTAL = bytes AVERAGE = avg.bytes ITEMS = count CKSUM = value BITS =
count

```

This message provides the following information for the records it has processed:

Parameter	Description
<i>field</i>	The name of the field for which the statistics are produced.
<i>filename</i>	The name of the file for which the statistics are produced.
<i>bytes</i>	The total number of bytes for values in the field.
<i>avg.bytes</i>	The average number of bytes of values in the field.

CHECK.SUM Message Output

Parameter	Description
<i>count</i>	The number of records processed.
<i>value</i>	The checksum value for the field.
<i>count</i>	The bit count of the values in the field.

CHECK.SUM Message Output (Continued)

The field mark value is calculated in the statistics.

The CHECK.SUM command can return misleading results if you compare the same data in two files, one in NLS mode, the other in non-NLS mode. To avoid comparing data in internal format in one file and external format in the other file, use the UNICODE.FILE command on one of the files before you use CHECK.SUM.

Examples

This example lists field statistics for all values in the field ZIP in the file SUN.MEMBER:

```
>CHECK.SUM SUN.MEMBER ZIP
```

```
Byte statistics for ZIP:
```

```
Total = 78 Average = 6 Items = 13 Cksum = 30020 Bits = 307
```

The next example lists statistics for the value in the field ZIP in record 7100:

```
>CHECK.SUM SUN.MEMBER ZIP "7100"
```

```
Byte statistics for ZIP:
```

```
Total = 6 Average = 6 Items = 1 Cksum = 2311 Bits = 23
```

CLEAN.ACCOUNT

Use CLEAN.ACCOUNT to perform routine system maintenance or to clear up suspected problems with the files in your account.

Syntax

CLEAN.ACCOUNT

Description

CLEAN.ACCOUNT looks in your account for specific system files and asks your permission to delete or clear the contents of these files. CLEAN.ACCOUNT also examines all the files in your account to verify that they are valid UniVerse files.

&TEMP& File

CLEAN.ACCOUNT looks for the file &TEMP&. This file is created by certain UniVerse commands that need a temporary file to store records. After you examine the records stored in &TEMP&, the file is usually no longer needed. If this file exists in your account, CLEAN.ACCOUNT asks if you want to delete it. Answer yes (**Y**) to delete the &TEMP& file.

&PH& File

CLEAN.ACCOUNT looks for the &PH& file. The **PHANTOM** command creates this file and uses it to store records containing data about each execution of the PHANTOM command. If you use the PHANTOM command often, this file can contain a large number of records. If this file exists, CLEAN.ACCOUNT asks if you want to clear this file. Answer yes (**Y**) to delete all the records in the &PH& file.

&SAVEDLISTS& File

CLEAN.ACCOUNT looks at the [&SAVEDLISTS&](#) file and deletes any records that begin with an ampersand (&). These records are temporary records, such as the temporary saved lists created by VVOC. A user's command stack history is also stored as a record in this file with a record ID that begins with an ampersand. After running CLEAN.ACCOUNT, the [&SAVEDLISTS&](#) file normally contains only lists created with the [SAVE.LIST](#) command.

UniVerse Files in Your Account Directory

CLEAN.ACCOUNT examines all the files in your account. For each file definition in the VOC file, CLEAN.ACCOUNT verifies that the files defined by the VOC record actually exist. If the file does not exist, an error message indicates the fact. CLEAN.ACCOUNT asks if you want to remove the path from field 2 of the VOC entry. If you answer yes (**Y**), the path in the VOC entry is set to an empty string. If the fields for the dictionary and the data file are set to empty strings in the VOC entry, the VOC entry is also deleted from the VOC file.

UniVerse Files in Other Account Directories

CLEAN.ACCOUNT looks at file definitions in the VOC file that use a path to point to files defined in other accounts. If the file does not exist, CLEAN.ACCOUNT asks if you want to remove the path from the file definition record. If you answer yes (**Y**), the path in the file definition record is set to an empty string. If the fields for the dictionary and the data file are set to empty strings in the VOC entry, the VOC entry is also deleted from the VOC file.

Files in Your Account Directory

CLEAN.ACCOUNT examines any files in your account directory that do not have a valid file definition in the VOC file. For any valid UniVerse file, CLEAN.ACCOUNT puts a file definition in the VOC file. A message appears indicating that a new file definition was added to the VOC file for the files. The names of any non-UniVerse files in your account are displayed with a message indicating that the files exist in your account but are not valid UniVerse files.

Example

```
1  >CLEAN.ACCOUNT
Do you wish to clear the file "&PH&" (Y/N) ? Y
Now clearing file "&PH&".
File "&PH&" has been cleared.

Now selecting all the File definition records in your VOC file.

Now selecting all the operating system files in your directory.

Now processing the VOC file definition records selected.

Processing file definition record  "NEWACC"
Processing file definition record  "ACCOUNTS"
Processing file definition record  "PAYABLES"
Processing file definition record  "SUN.MEMBERS"
Processing file definition record  "VOC"
Processing file definition record  "BP.O"
Processing file definition record  "BP"
>
```

CLEAR.FILE

Use CLEAR.FILE to remove all records from a file.

Syntax

CLEAR.FILE [DICT | DATA] [*filename*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies that you want to clear the file dictionary. If you do not specify DICT, the data file is cleared.
DATA	Specifies that you want to clear the data file specified by <i>filename</i> . DATA is the default.
<i>filename</i>	Specifies the name of the file to be cleared. If you do not specify <i>filename</i> and there is no active select list, CLEAR.FILE prompts for the file name. If you do not specify <i>filename</i> and there is an active select list, CLEAR.FILE clears the files named by the select list. If you specify <i>filename</i> and there is an active select list, CLEAR.FILE clears <i>filename</i> and the files named in the select list.

CLEAR.FILE Parameters

Description

If you use a select list with CLEAR.FILE, the files named in the select list must be defined in the VOC file. If *filename* specifies a VOC entry that is a file synonym or a remote file entry, CLEAR.FILE clears the file specified in that VOC entry.

Once the CLEAR.FILE process has begun, you cannot interrupt it by pressing the Break key.

CLEAR.FILE does not remove the file itself. To remove the file, use the [DELETE.FILE](#) command.

Examples

This example clears the data file PAYABLES:

```
>CLEAR.FILE PAYABLES  
File "PAYABLES" has been cleared.
```

The next example clears the dictionary of PAYABLES:

```
>CLEAR.FILE DICT PAYABLES  
File "DICT PAYABLES" has been cleared.
```

CLEAR.LOCKS

Use CLEAR.LOCKS to release either a specified task synchronization lock or all task synchronization locks that were set by your task. UniVerse has 64 semaphores called task synchronization locks that synchronize multiple processes running at the same time in the same account or in different accounts.

Syntax

CLEAR.LOCKS [*n*]

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>n</i>	The number of the lock you want to release. If you do not specify a lock number, CLEAR.LOCKS clears all the locks set by your task.

CLEAR.LOCKS Parameter

Description

Before using CLEAR.LOCKS, be sure you know how the locks have been set for your installation. Consult your UniVerse administrator if you are not sure. If you release a lock that should be retained, you might damage the data. CLEAR.LOCKS does not check the current execution status of a lock. It verifies only that your task set the lock.

Example

In this example the user first sets lock 5 with the LOCK command. The LIST.LOCKS report shows that user 34 set lock 5.

```
>LOCK 5
>LIST.LOCKS
 0:----- 1:----- 2:----- 3:----- 4:----- 5:34      6:-----
 7:-----
 8:----- 9:----- 10:----- 11:----- 12:----- 13:----- 14:-----
15:-----
16:----- 17:----- 18:----- 19:----- 20:----- 21:----- 22:-----
23:-----
24:----- 25:----- 26:----- 27:----- 28:----- 29:----- 30:-----
31:-----
32:----- 33:----- 34:----- 35:----- 36:----- 37:----- 38:-----
39:-----
40:----- 41:----- 42:----- 43:----- 44:----- 45:----- 46:-----
47:-----
48:----- 49:----- 50:----- 51:----- 52:----- 53:----- 54:-----
55:-----
56:----- 57:----- 58:----- 59:----- 60:----- 61:----- 62:-----
63:-----
```

Next, lock 5 is cleared:

```
>CLEAR.LOCKS 5
Cleared LOCK number 5.
>LIST.LOCKS
 0:----- 1:----- 2:----- 3:----- 4:----- 5:----- 6:-----
 7:-----
 8:----- 9:----- 10:----- 11:----- 12:----- 13:----- 14:-----
15:-----
16:----- 17:----- 18:----- 19:----- 20:----- 21:----- 22:-----
23:-----
24:----- 25:----- 26:----- 27:----- 28:----- 29:----- 30:-----
31:-----
32:----- 33:----- 34:----- 35:----- 36:----- 37:----- 38:-----
39:-----
40:----- 41:----- 42:----- 43:----- 44:----- 45:----- 46:-----
47:-----
48:----- 49:----- 50:----- 51:----- 52:----- 53:----- 54:-----
55:-----
56:----- 57:----- 58:----- 59:----- 60:----- 61:----- 62:-----
63:-----
```

CLEARCOMMON

Use CLEARCOMMON to reset the values of all the variables in the common area to zero. Variables outside the common area are unaffected.

Syntax

CLEARCOMMON

Description

The BASIC COMMON statement assigns common variables.

CLEARCOMMON sets to zero any common area variables that are not assigned.

Unnamed common variables are cleared when the UniVerse BASIC program that assigned them completes execution. All common variables are cleared when you exit UniVerse.

CLEARDATA

Use CLEARDATA to clear the data stack built either from DATA statements in a paragraph or from DATA statements in a BASIC program. The CLEARDATA command does the same thing as the BASIC CLEARDATA statement.

Syntax

CLEARDATA

Description

Use CLEARDATA when a data stack exists and a BASIC program requests input from the terminal. CLEARDATA is useful for clearing data when an error occurs in processing DATA statements.

In a BASIC program, use CLEARDATA as part of an EXECUTE statement.

When a BASIC program returns control to the command processor, the data stack is cleared.

Example

This program shows an example of the CLEARDATA command:

```
1 10 *
PRINT 'Enter filename':
INPUT FILE
PRINT 'Enter record id':
INPUT ID
PRINT 'Enter procedure':
INPUT PROC
OPEN ' ', FILE TO FILE.NAME ELSE PRINT 'CANNOT OPEN FILE':FILE
EXECUTE 'CLEARDATA'
GOTO 10
END
```

CLEARPROMPTS

Use CLEARPROMPTS in a paragraph to set the value of in-line prompts to empty strings.

Syntax

CLEARPROMPTS

Description

Once a value has been entered at an inline prompt, the prompt retains that value until a CLEARPROMPTS command is executed, unless the inline prompt control option A is specified. CLEARPROMPTS clears all values entered for in-line prompts.

For information about in-line prompts, see the <<...>> command.

Example

The following paragraph prompts for a new value each time the loop is repeated. If the CLEARPROMPTS statement is omitted, the loop repeats endlessly using the value for <<ENTER MEMBERSHIP.NBR>> entered at the initial prompt.

```
1          LIST.MEMBERS
0001: PA
0002: LOOP
0003: CLEARPROMPTS
0004: IF <<ENTER MEMBERSHIP.NBR>> EQ 'END' GO END:
0005: LIST SUN.MEMBER WITH @ID EQ <<ENTER MEMBERSHIP.NBR>>
0006: REPEAT
0007: END:
```

CLEARSELECT

Use CLEARSELECT to cancel an active select list.

Syntax

CLEARSELECT [*list#* | ALL]

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>list#</i>	The numbered select list you want to clear. The list number can be from 0 through 10, or it can be ALL, which clears all select lists. If you omit the list number, CLEARSELECT clears select list 0.

CLEARSELECT Parameter

Description

Use CLEARSELECT when you have created an active select list and you decide you do not want to use the list. You can clear an entire select list or the unused remainder of an active select list.

Example

This example clears select list 5:

```
>SSELECT INVENTORY WITH COST > 9 TO 5  
  
10 record(s) selected to SELECT list #5.  
>CLEARSELECT 5  
SELECT list number 5 cleared.
```

CLR

Use CLR to clear the screen and return the cursor to the home position. CLR is a synonym for the [CS](#) command.

Syntax

CLR

CNAME

Use CNAME to change the name of a file or to change the names of records in a file.

Syntax

CNAME *old.filename* { TO | , } *new.filename*

CNAME [DICT] *filename old* { TO | , } *new*

CNAME [DICT] *filename old, new* [*old2, new2*] ...

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
<i>filename</i>	A name of a file. <i>old.filename</i> is the filename to be changed, <i>new.filename</i> is the new name of the file.
<i>old</i>	The name of the record you want to change.
<i>new</i>	The new name of the record.

CNAME Parameters

Description

When you change a filename or record ID name, a message confirms the change.

Changing Filenames

To change the name of a file, you must specify the old filename, either the keyword TO or a comma, then the new filename.

To change the name of a data file that is one of multiple data files sharing the same file dictionary, you must use the following syntax:

CNAME *filename,old.datafile* TO *filename,new.datafile*

CNAME changes the record ID of the file definition in the VOC file to the new file name. It also changes the paths in the VOC entry in the following way:

- If the path in field 2 of the VOC entry is *old.filename*, CNAME changes the name of the file or directory and also changes the path in field 2 to *new.filename*. If the path in field 2 is not *old.filename*, CNAME changes nothing.
- If the path in field 3 of the VOC entry is *D_old.filename*, CNAME changes the file name and the path in field 3 to *D_new.filename*. If the path in field 3 is not *D_old.filename*, CNAME changes nothing.

Changing Record Names

To change record IDs in a file, specify *filename* followed by the old and new record IDs. To change the record ID of a record in the file dictionary, use the DICT keyword before file name. If you are changing just one record ID, you can specify *old* and *new* separated by either a comma or the keyword TO. If you want to change more than one record ID, separate *old* and *new* by a comma. You must separate the *old*, *new* pairs from each other with spaces.

Restrictions

You cannot use CNAME to change the name of an SQL table.

You cannot use CNAME to change the name of a part file belonging to a distributed file.

Examples

The original VOC file entry looks like this:

```
PAYABLES
0001 F
0002 PAYABLES
0003 D_PAYABLES
```

When you enter the following command:

>CNAME PAYABLES TO ACCT_PAYABLES

the following messages appear:

```
Changed operating system file name from "PAYABLES" to "ACCT_PAYA000"  
Changed operating system file name from "D_PAYABLES" to  
"D_ACCT_PAYA000"  
Changed "PAYABLES" to "ACCT_PAYABLES" in your VOC file.
```

The new VOC file entry looks like this:

```
      ACCT_PAYABLES  
0001  F  
0002  ACCT_PAYA000  
0003  D_ACCT_PAYA000
```

The next example changes record 1234 to 123400 in the PAYABLES file:

>CNAME PAYABLES 1234, 123400

COMO

Use COMO to start or stop copying terminal output to a record in the **&COMO&** file. You can also use COMO to print records from the **&COMO&** file, delete them from the system, or list their names.

Syntax

COMO [*action*]

Parameters

The following table describes the valid values for *action*.

Action	Description
ON <i>record</i> [HUSH]	Creates a COMO record and starts copying terminal output to it. HUSH suppresses terminal display. Output is copied to <i>record</i> but is not displayed on the screen. Be careful. Because prompts for input are not displayed, your program or paragraph can wait indefinitely for input.
OFF	Stops copying terminal output to the COMO record.
DELETE { <i>record</i> * }	Deletes a COMO record. Asterisk (*) clears the &COMO& file.
LIST	Lists COMO records in your account.
SPOOL <i>record</i> [T]	Prints a COMO record on the line printer. T displays a COMO record on the screen.

COMO Actions

Description

COMO is short for command output.

If you use COMO without any options, COMO prompts for the necessary information.

COMO records are records in a type 1 file named **&COMO&**. If you create a COMO record by using COMO ON, the COMO command creates an **&COMO&** file if one does not exist in your account. In a COMO ON statement, if you specify the name of an **&COMO&** record that exists, the old contents of the record are overwritten. The records produced by COMO can be quite large. Remember to delete COMO records when you no longer need them (use COMO DELETE *record*).

The **TANDEM** command cannot capture output directed to an **&COMO&** file.

Examples

In the following example, COMO is used with no options. It establishes a COMO record of the terminal session. Everything that appears on the terminal is stored in the record SESSION1 in the file **&COMO&** until the command COMO OFF is issued. Terminal display is on, so output appears on the screen at the same time as it is copied to the COMO record.

```
>COMO
Enter action (ON, OFF, DELETE, LIST, SPOOL, QUIT) = ON
Enter COMO file name = SESSION1
Terminal display on? (Y/N) = Y
```

The next example lists all COMO records in the **&COMO&** file. The COMO record SPOOLLIST is deleted, and the user is prompted for the next action. QUIT returns the user to the system prompt.

```
>COMO LIST
COMO file listing 02:22:49pm 02 Jul 1995

01 SPOOLLIST
02 FILESTATS
03 DICTVOC
Enter number of file to be selected = 1
Enter action (ON, OFF, DELETE, LIST, SPOOL, QUIT) = DELETE
COMO SPOOLLIST deleted.
Enter action (ON, OFF, DELETE, LIST, SPOOL, QUIT) = QUIT
>
```

COMPILE.DICT

Use COMPILE.DICT to compile the I-descriptors in a file dictionary before using them in a sentence. You can compile one, several, or all of the I-descriptors in the dictionary.

Syntax

COMPILE.DICT *filename* [*descriptors*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>filename</i>	The name of the UniVerse file whose dictionary entries are to be compiled. Do not specify the DICT keyword.
<i>descriptors</i>	The names of the I-descriptors to be compiled. If you do not specify a descriptor name, all I-descriptors in the dictionary are compiled. You can also use an active select list to specify I-descriptors.

COMPILE.DICT Parameters

Description

COMPILE.DICT stores the compiled object code and the time and date of the compilation in the I-descriptor record. If you change the I-type expression, the Editor flags the I-descriptor and displays a reminder that the I-descriptor must be recompiled when you file the record. It invalidates the existing descriptor as well. Retrieve compiles this I-descriptor before using it in a sentence. Because the I-descriptors in a dictionary are often related, they should always be compiled together.

COMPILE.DICT is a synonym for the [CD](#) command.

Example

This example compiles four I-descriptors:

```
>COMPILE.DICT ORDERS
Compiling "*A9998".
@RECCOUNT
Compiling "EXT".
QTY * ( TRANS ( BOOKS , CODE , PRICE , X ) )
Compiling "PRICE".
TRANS ( BOOKS , CODE , PRICE , X )
Compiling "TITLE".
TRANS ( BOOKS , CODE , TITLE , X )
```

COMPILE.DICTS

Use COMPILE.DICTS to compile all the dictionaries in an account.

Syntax

COMPILE.DICTS

Description

COMPILE.DICTS runs the COMPILE.DICT command on every file in the account. The output of COMPILE.DICTS is also saved in a record of &COMO& called COMPILE.DICTS. For more information about the [&COMO&](#) file, see the [COMO](#) command.

CONFIG

Use CONFIG to display the currently active authorization parameters and current configurable parameter values.

Syntax

CONFIG [ALL | BRIEF | DATA]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
ALL	Lists the information from the BRIEF and DATA keywords.
BRIEF	(Default) Lists the license number, the licensed number of users for the system, and the license expiration date.
DATA	Lists the current values of the UniVerse configurable parameters.

CONFIG Parameters

Example

```
>CONFIG ALL
Configuration data for license number 1:
User limit =      64
Expiry date=    1/2/2032
-UVNET          package is installed.
                  User limit =      64. Expiry date = 1/2/2032
-GCI            package is installed.
                  User limit =      64. Expiry date = 1/2/2032
-NLS            package is installed.
                  User limit =      64. Expiry date = 1/2/2032
-UCI            package is installed.
                  User limit =       1. Expiry date = 1/2/2032
-UVCS            package is installed.
                  User limit =      64. Expiry date = 1/2/2032
-UVADM           package is installed.
                  User limit =      64. Expiry date = 1/2/2032
Current tunable parameter settings:
```

```

MFILES = 12
T30FILE= 200
OPENCHK= 1
WIDE0 = 0x3dc00000
UVSPOOL= /usr/spool/uv
UVTEMP = /tmp
SCRMIN = 3
SCRMX = 5
SCRSIZE= 512
QDEPTH = 16
HISTSTK= 99
QSRUNSZ= 2000
QSB RNCH= 4
QSDEPTH= 8
QSMXKEY= 32
TXMODE = 0
LOGBLSZ= 512
LOGBLNUM= 8
LOGSYCNT= 0
LOGSYINT= 0
TXMEM = 32
OPTMEM = 64
SELBUF = 4
ULIMIT = 128000
FSEMNUM= 23
GSEMNUM= 23
PSEMNUM= 64
FLTABSZ= 11
GLTABSZ= 300
RLTABSZ= 300
RLOWNER= 300
PAKTIME= 300
NETTIME= 5
QBREAK = 1
VDIVDEF= 1
UVSYNC = 1
BLKMAX = 8192
PICKNULL= 0
SYNCALOC= 1
MAXRLOCK= 100
ISOMODE = 1
PKRJUS T = 0
PROCACMD= 0
PROCRCMD= 0
PROCPRMT= 0
ALLOWNFS= 0
CSDHDISPATCH = /usr/bin/csh
SHDISPATCH = /usr/bin/sh
DOSDISPATCH = NOT_SUPPORTED
NLSMODE = 1
NLSREADELSE = 1
NLSWRITEELSE = 1
NLSDEFFILEMAP = ISO8859-1+MARKS
NLSDEFDIRMAP = ISO8859-1+MARKS

```

```
NLSNEWFILEMAP = NONE
NLSNEWDIRMAP = ISO8859-1+MARKS
NLSDEFPTRMAP = ISO8859-1+MARKS
NLSDEFTERMMAP = ISO8859-1+MARKS
NLSDEFDEVMAP = ISO8859-1+MARKS
NLSDEFGCIMAP = ISO8859-1+MARKS
NLSDEFSRVMAP = ISO8859-1+MARKS
NLSDEFSEQMAP = ISO8859-1+MARKS
NLSOSMAP = ISO8859-1
NLSLCMODE = 1
NLSDEFUSERLC = OFF
NLSDEFSRVLC = OFF
LAYERSEL= 0
OCVDATE= 0
MODFPTRS= 1
THDR512= 0
UDRMODE= 0
UDRBLKS= 0
AFFNMODE= 0
```

CONFIGURE.FILE

Use CONFIGURE.FILE to change the parameters of an existing dynamic file.

Syntax

CONFIGURE.FILE [DICT] [*filename*] [*parameter* [*value*] ... | DEFAULTS]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
<i>filename</i>	The name of the dynamic file to be reconfigured.

CONFIGURE.FILE Parameters

parameter specifies new parameters for the dynamic file. Parameters not specified are unchanged. *parameter* can be any of the following:

Parameter	Description
MINIMUM.MODULUS <i>n</i>	Specifies the minimum modulo of the file. This keyword takes an integer argument greater than 1. This value is also the initial value of the modulo of the dynamic file. A minimum modulo of 1 is the default.
SPLIT.LOAD <i>n</i>	Specifies the level at which the file’s modulo is increased by 1. SPLIT.LOAD takes a numeric argument indicating a percentage of the space allocated for the file. When the data in the file exceeds the space allocated for the file, the data in one of the groups divides equally between itself and a new group, to increase the modulo by one. The default SPLIT.LOAD is 80%.
MERGE.LOAD <i>n</i>	Specifies the level at which the file’s modulo is decreased by 1. MERGE.LOAD takes a numeric argument indicating a percentage of the space allocated for the file. When the data in the file uses less than the space allocated for the file, the data in the last group of the file merges with another group, to decrease the modulo by 1. The default MERGE.LOAD is 50%.
Dynamic File Parameters	

Parameter	Description
LARGE.RECORD <i>n</i>	Specifies the size of a record too large to be included in the primary group buffer. LARGE.RECORD takes an argument that can be specified as an integer or a percentage. Specified as an integer, the argument is the number of bytes a record must contain to be considered a large record. Specified as a percentage, the argument is a percentage of the group size. When the size of a record exceeds the value specified, CONFIGURE.FILE puts the data for the record in an overflow buffer, but puts the record ID in the primary buffer. This method of large record storage increases access speed. The default LARGE.RECORD size is 80%.
MINIMIZE.SPACE	Causes the values for the split load, merge load, and the large record size to be calculated to optimize the amount of space required by the file at the expense of access time. If you specify a value for split load, merge load, or large record size, the value you specify overrides the value calculated by MINIMIZE.SPACE.
DEFAULTS	Sets all the parameters of the specified dynamic file to the default values.

Dynamic File Parameters (Continued)

Description

Be careful when changing dynamic file parameters. The default parameters are set so most dynamic files work correctly. For some files, you can increase efficiency by changing the default file parameters.

We recommend that you back up your files.

Always verify that the file’s new parameters are more efficient than the old ones. Use the [ANALYZE.FILE](#) command to verify the new parameters after you change them.

You cannot use CONFIGURE.FILE to change the values of GROUP.SIZE, RECORD.SIZE, or the hashing algorithm. To change these values, use the [CREATE.FILE](#) command to create a new dynamic file with the parameters you want, then copy the contents of the old file to your newly created file using the [COPY](#) command.

Examples

This example reconfigures the file FILMS to minimize the disk space required. The MINIMIZE.SPACE option calculates new values for the split and merge loads and the large record size, and changes the values accordingly.

```
>CONFIGURE.FILE FILMS MINIMIZE.SPACE
File name                               = FILMS
Changing Split Load from 80% to 180%.
Changing Merge Load from 50% to 80%.
Changing Large Record Size from 1628 bytes to 2036 bytes.
File name                               = FILMS configured.
>
```

In the next example, the [ANALYZE.FILE](#) command lists the current parameter values of the CUSTOMERS file:

```
>ANALYZE.FILE CUSTOMERS
File name ..... CUSTOMERS
Pathname ..... CUSTOMERS
File type ..... DYNAMIC
Hashing Algorithm ..... GENERAL
No. of groups (modulus) .... 12 current ( minimum 1 )
Large record size ..... 1628 bytes
Group size ..... 2048 bytes
Load factors ..... 80% (split), 50% (merge) and 61%
(actual)
Total size ..... 6144 bytes
>
```

Because CUSTOMERS is expected to grow, the next example uses CONFIGURE.FILE to change the large record size and the minimum modulus:

```
>CONFIGURE.FILE CUSTOMERS LARGE.RECORD 1200 MINIMUM.MODULUS 200
File name                               = CUSTOMERS
Changing Minimum Modulus from 1 to 200.
Changing Large Record Size from 1628 bytes to 1200 bytes.
File name                               = CUSTOMERS configured.
>
```

Another [ANALYZE.FILE](#) verifies the new parameter values:

```
>ANALYZE.FILE CUSTOMERS
File name ..... CUSTOMERS
Pathname ..... CUSTOMERS
File type ..... DYNAMIC
Hashing Algorithm ..... GENERAL
No. of groups (modulus) .... 12 current ( minimum 200 )
```



```
Large record size ..... 1200 bytes
Group size ..... 2048 bytes
Load factors ..... 80% (split), 50% (merge) and 61%
(actual)
Total size ..... 6144 bytes
>
```

CONNECT

Use CONNECT to connect to a local or remote SQL server, such as another UniVerse system or an ORACLE, SYBASE, or DB2 system.

Syntax

CONNECT *data.source* [*option setting* [*option setting*] ...]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>data.source</i>	The name of the data source to which you want to connect. The data source must be defined in the <i>uvodbc.config</i> file. If you do not enter the name of a data source, CONNECT lists all data sources in the <i>uvodbc.config</i> file.
<i>option</i>	One of the following options to control the input format or output display: BLOCK PREFIX INVERT UVOUT MVDISPLAY VERBOSE NULL WIDTH Specify any option by typing the word or its first letter. Each option must be followed by <i>setting</i> .
<i>setting</i>	The new setting for the option.

CONNECT Parameters

The following sections describe each option and its possible settings in detail.

BLOCK Option.

This option defines how input statements will be terminated. *setting* is one of the following:

Value	Description
ON	Enables block mode. In this mode, you can enter a series of SQL statements, ending each with a semicolon(;). To terminate the block of SQL statements, press ENTER immediately after an “SQL+” prompt.
OFF	(Default) Disables block mode. In this mode, if you enter a semicolon at the end of a line of input, the SQL Client terminates your input and sends it to the data source.
<i>string</i>	Enables block mode (see ON). <i>string</i> must be from one to four characters. To terminate the block of SQL statements, enter <i>string</i> immediately after an “SQL+” prompt.

BLOCK Option Settings

INVERT Option.

This option lets you control case inversion for alphabetic characters you type while CONNECT is running. *setting* is one of the following:

Value	Description
ON	Inverts the case of all alphabetic characters you type — that is, lowercase letters change to uppercase, and uppercase letters change to lowercase. This is equivalent to setting PTERM CASE parameters to INVERT and LC-IN.
OFF	Disables case inversion. This is equivalent to setting PTERM CASE parameters to NOINVERT and LC-IN. This is the default setting for non-UniVerse data sources.
INIT	Sets case-inversion parameters to the values they had when you invoked CONNECT. This is the default setting for UniVerse data sources.

INVERT Option Settings

When you exit from CONNECT, case inversion for input is restored to the state it was in when you invoked CONNECT.

MVDISPLAY Option

The MVDISPLAY option defines how to display value marks in multivalued data (when connected to a UniVerse data source). For each row, multiple values in the same field are displayed on the same line, separated by value marks. *setting* is one of the following:

Value	Description
SPACE	Displays a value mark as a blank space.
NOCONV	Displays a value mark as CHAR(253).
<i>char</i>	Displays a value mark as <i>char</i> (one character).

MVDISPLAY Option Settings

By default, value marks are displayed as * (asterisk).

NULL Option

The NULL option defines how to display the SQL null value. *setting* is one of the following:

Value	Description
SPACE	Displays SQL null as a blank space.
NOCONV	Displays SQL null as CHAR(128).
<i>string</i>	Displays SQL null as <i>string</i> . The string can be from one to four characters. By default, null is displayed as the four-character string NULL.

NULL Option Values

PREFIX Option

The PREFIX option defines the prefix character for local commands. *setting* is any valid prefix character. The default prefix character is a period (.). You can use only the following characters as the prefix character:

!	exclamation point	?	question mark
@	at sign	(	left parenthesis
#	hash sign	)	right parenthesis
\$	dollar sign	{	left brace
%	percent	}	right brace
&	ampersand	[	left bracket
*	asterisk	]	right bracket
/	slash	‘	left quotation mark
\	backslash	’	right quotation mark
:	colon	.	period
=	equal sign		vertical bar
+	plus sign	"	double quotation mark
–	minus sign	,	comma

Valid Prefix Characters

UVOUT Option

The UVOUT option specifies how to handle output from SELECT commands executed on the data source. *setting* is either:

Value	Description
<i>filename</i>	Stores output in <i>filename</i> on the client, then displays the output from <i>filename</i> . If the file does not exist, the CONNECT command creates it.
OFF	(Default) Displays output from the data source directly on the client's screen.

UVOUT Option Settings

VERBOSE Option

The VERBOSE option displays extended column information and system messages. *setting* is one of the following:

Value	Description
ON	Enables verbose mode. In this mode, the name, SQL data type, precision, scale, and display size are displayed for each column's definition when selecting data from the data source. Error messages are displayed in extended format that includes the type of call issued, status, SQLSTATE, error code generated by the data source, and the complete error text.
OFF	(Default) Disables verbose mode.

VERBOSE Option Settings

WIDTH Option

The WIDTH option defines the width of display columns. *setting* is one of the following:

Value	Description
<i>col#,width</i>	Sets the width of column <i>col#</i> to <i>width</i> . Do not enter a space after the comma. Specify <i>col#</i> as an asterisk (*) to set the width of all columns. <i>width</i> can be from 4 to the maximum line length allowed by your terminal. The default width for all columns is 10.
T	(Default) Truncates data that is wider than the specified <i>width</i> .
F	Folds data that is wider than the specified <i>width</i> onto multiple lines.
?	Displays the current column width settings, and tells whether data will be truncated or folded.

WIDTH Option Settings

Description

The CONNECT command lets you submit SQL statements to a specified data source and receive results at your terminal.

While you are connected to a data source, you can type any SQL statement understood by the data source's DBMS engine, including SELECT, INSERT, UPDATE, DELETE, GRANT, and CREATE TABLE. You cannot, however, successfully control statements such as BEGIN TRANSACTION, COMMIT, and ROLLBACK when you are using CONNECT.

If you execute a stored procedure or enter a command batch with multiple SELECT statements, the results of only the first SELECT statement are returned.

UniVerse and SYBASE treat SQL identifiers and keywords case-sensitively, whereas ORACLE and INFORMIX do not. UniVerse requires you to specify all keywords in uppercase; SYBASE requires you to specify data types (char, int, float, and so forth) in lowercase. In ORACLE and INFORMIX you can use either upper- or lowercase letters for these keywords.

CONVERT.ACCOUNT

Use CONVERT.ACCOUNT to convert a Pick or Prime INFORMATION account from its original format to a format compatible with UniVerse.

Syntax

CONVERT.ACCOUNT

Description

CONVERT.ACCOUNT invokes the account conversion menu. The menu comprises two submenus, one for converting Pick and Prime file dictionaries (including the VOC file), the other for converting BASIC programs. The dictionary conversion menu also has an option for converting REALITY procs to a proc format compatible with UniVerse.

For more information about converting Pick and Prime INFORMATION accounts, see the *UniVerse Guide for Pick Users*.

CONVERT.SQL

Use CONVERT.SQL to convert a UniVerse file to a table. You must be an SQL user to run CONVERT.SQL, and you must run it in an SQL schema. You must have write permissions for the directory and the VOC file in the account.

Syntax

CONVERT.SQL [*filename* [*action* [*options*]]]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>filename</i>	Specifies the UniVerse file to be converted. If <i>filename</i> contains special characters, enclose it in double quotation marks. <i>filename</i> must be an F- or Q-pointer in the VOC file of the schema you are logged in to. <i>filename</i> cannot be a type 1 or type 19 file, a distributed file, or a file comprising multiple data files. If you do not specify <i>filename</i> , you are prompted for it.

CONVERT.SQL Parameters

action is one of the following:

Action	Description
CREATE	Analyzes the file dictionary or the SQLDEF file, lists column and association definitions, and converts the file to a table. You can use any of the <i>options</i> with the CREATE action.
INFO	Lists D-, A-, and S-descriptors in the dictionary of <i>filename</i> .

CONVERT.SQL Actions

Action	Description
RESTORE	Restores a previously converted table to a UniVerse file, leaving the data in its current state. Also restores the table dictionary to its original contents.
RESTOREDATA	Restores a previously converted table to a UniVerse file, restoring both the data file and the dictionary to their original contents. This action works only if the contents of the data file were previously saved using the SAVEDATA option.
TEST	Analyzes the file dictionary or the SQLDEF file and lists column and association definitions. It does not convert the file to a table. You can use any of the <i>options</i> except SAVEDATA with the TEST action.

CONVERT.SQL Actions (Continued)

options are one or more of the following:

Option	Description
GEN	Generates a new SQLDEF file even if an old one exists. Use this option only with the TEST or CREATE action.
SHOW	Displays the generated CREATE TABLE statement. Use this option only with the TEST or CREATE action.
BRIEF	Suppresses the list of column and association definitions. Use this option only with the TEST or CREATE action.
SAVEDATA	Saves the contents of the original UniVerse data file as <i>filename_SQLSAVE</i> . Use this option only with the CREATE action.
LPTR	Sends output to the printer.

CONVERT.SQL Options

Description

CONVERT.SQL analyzes the file's dictionary and generates an intermediate file called *filename_SQLDEF*, known as the SQLDEF file. The SQLDEF file contains column and association definitions based on the contents of the file dictionary. The information in the SQLDEF file is used to form a CREATE TABLE statement which is executed to convert the file to a table. This process adds a SICA (security and integrity constraints area) to the file's header, adds SQL datatype codes to the file's dictionary, and updates the SQL catalog with information about this new table.

Neither the data file nor the file dictionary need be in the directory you are logged in to, but the VOC file must contain an F- or Q-pointer to both.

You can use CONVERT.SQL interactively, or you can specify an action you want the command to perform automatically.

Using CONVERT.SQL Interactively

If you specify no arguments on the command line, CONVERT.SQL prompts you to enter a filename. If you specify only a file name, either on the command line or in response to the prompt, CONVERT.SQL operates interactively.

If an SQLDEF file exists, you are prompted to use it or to generate a new one. If there is no SQLDEF file, the program generates one based on an analysis of the file dictionary. It displays a list of columns and prompts you to edit them or take some other action (see [“Editing Column and Association Definitions”](#) on page 136).

In this mode, CONVERT.SQL does the following:

- Analyzes the D-, A-, S-descriptors and PH entries in the file dictionary.
- Generates proposed column and association definitions and stores them in a file called *filename*_SQLDEF. Or you can use an existing SQLDEF file.
- Displays the SQL column and association definitions in the SQLDEF file.
- Displays the interactive editing prompt, which lets you modify the proposed table definition.
- Displays the generated CREATE TABLE statement, if requested.
- Executes the CREATE TABLE statement, if requested. This adds SICA information to the file header and updates the SQL catalog.

You can leave interactive mode by requesting that the file be converted to a table (using the current set of column definitions) or by typing **Q**.

Using CONVERT.SQL Automatically

If you specify an *action* on the command line, CONVERT.SQL performs the action automatically, with no user intervention. Depending on what action you specify, CONVERT.SQL performs one or more of the operations described in the previous section.

Generating Column and Association Definitions

The generation phase creates the SQLDEF file which contains valid SQL definitions for columns and associations of this table. (The SQLDEF file is not deleted automatically.)

Column Definitions

During this phase, CONVERT.SQL creates column definitions as follows. A maximum of 500 columns plus the primary key are allowed.

1. A column definition is generated for each D-, A-, and S-descriptor in the dictionary, with the following exceptions:
 - A- and S-descriptors that specify a correlative code (field 8) are ignored.
 - Descriptors whose names start with @Ak. (from a BUILD.INDEX process) are ignored.
 - Descriptors with a nonnumeric location (field 2), or with a location greater than 500, are ignored.
 - Additional column definitions are generated if sequential numbers for data fields do not exist. For example, if the dictionary contains definitions for fields 1, 2, and 4, CONVERT.SQL also generates a column definition for column 3.
2. CONVERT.SQL encloses all field names in double quotation marks in order to preserve field names with special characters and to allow field names which are reserved words.
3. An SQL data type is chosen for each column definition (see [“Data Types”](#) on page 134).

4. If field synonyms exist in the dictionary, each column definition is assigned an alphabetic designator (uppercase or lowercase) appended to the field number. The preferred column definition is assigned the character A; synonyms are assigned B, C, and so forth. Each column can have a maximum of 50 synonyms.

CONVERT.SQL chooses the preferred column definition from the synonyms by applying the following criteria in the order shown:

- *fieldname* is in the @SELECT phrase.
 - *fieldname* is a legal column name (as a delimited identifier) for the operating system.
 - A multivalued field in an association is preferred to an unassociated multivalued field, which is preferred to a singlevalued field.
 - *fieldname* dictionary entry specifies the SQL data type.
 - *fieldname* is in the @KEY phrase.
 - *fieldname* is in the @ phrase.
 - *fieldname* is not a number.
 - *fieldname* is not @ID.
5. If the dictionary contains an @KEY phrase naming more than one field, each of which is an I-descriptor or defines a correlative (A- or S-type), CONVERT.SQL creates a multicolumn primary key for the table. If the dictionary also contains an @KEY_SEPARATOR entry, CONVERT.SQL specifies the character it defines as the separator of the columns in the primary key.

Note: Be sure that the I-descriptors and correlative definitions named in the @KEY phrase properly extract the key columns from the record ID.

Data Types

Each column's data type is determined by the SQLTYPE, conversion, and justification as follows:

- If the dictionary definition specifies an SQLTYPE, this is used for the data type.

- If the dictionary does not specify SQLTYPE but does specify a conversion code, the data type is determined from the conversion code as shown in the following table:

This conversion code...	Generates this SQL date type...
MD, ML, or MR with nonzero scale	DEC (9, <i>s</i>) where <i>s</i> is the scale determined from the conversion code
MD, ML, or MR with scale = zero or none	INT
Q	REAL
D	DATE
MT	TIME
MB, MO, MX, or NR	INT
BB or BX	VARBIT
All others	VARCHAR

Determining the Data Type

- If the dictionary does not specify SQLTYPE or conversion code, the data type is determined from the justification as follows:

This justification...	Generates this SQL data type...
T or none	VARCHAR
Q, QR, or QL	REAL
R	INT
Other	CHAR (<i>n</i>) or VARCHAR(<i>n</i>) where <i>n</i> is the field's width.

Determining Data Type from Justification Code

@SELECT Phrase

If the file dictionary does not contain an @SELECT phrase, CONVERT.SQL generates one for the table, based on the dictionary's @ phrase. It includes any I-descriptors, as well as A- and S-descriptors that define correlatives in field 8, that appear in the @ phrase. If there is no @ phrase, CONVERT.SQL creates an @SELECT phrase naming all of the table's columns.

Association Definitions

CONVERT.SQL creates SQL association definitions from the following:

- Information in the ASSOC field of the D-, A-, and S-descriptors
- A dictionary's @ASSOC_KEY.multivaluename X-descriptors

A maximum of 49 associations are allowed.

Associations are created from the ASSOC field as follows:

1. If an association has no @ASSOC_KEY.multivaluename X-descriptor, the association is defined as INSERT LAST and has no association keys.
2. @DC n is the name given to associations that are generated from A- or S-descriptors. n is the location of the data field whose dictionary entry contains a C code in the ASSOC field.
3. If the same field number appears in more than one association definition, CONVERT.SQL designates the duplicates as overlapping associations and eliminates the duplicates.

Associations created from @ASSOC_KEY.multivaluename depend on the content of field 2 of the X-descriptor. They are created as follows:

1. If field 2 contains UNSTABLE, the association is defined as INSERT LAST and has no association key.
2. If field 2 contains STABLE, the association is defined as INSERT PRESERVING and has no association key.
3. If field 2 contains KEY columns, the association is defined as having the specified association key columns.

List Column and Association Definitions

At the end of the generation phase, CONVERT.SQL displays the list of column definitions (including synonyms), and association definitions (including overlapping associations).

Editing Column and Association Definitions

If you are running CONVERT.SQL interactively, after the generation phase you can edit the SQL column and association definitions. The following prompt appears:

Enter C..., D..., U..., R..., R, S, X, Q, or H for Help [R]:

Choose one of the following options:

Value	Description
<i>Ctns/old/new</i> [/G]	Changes the column or association definition. The new definition is added, the old definition becomes a synonym. You can also use the backslash (\) as a delimiter. <i>t</i> C or empty for a column, A for an association, or K for key part. <i>n</i> 1-, 2-, or 3-digit column, association, or key part number. <i>s</i> Optional spaces.
<i>CtnsTs type</i>	Changes the data type of a column definition to <i>type</i> . <i>t</i> C or empty for a column, or K for key part. <i>n</i> 1-, 2-, or 3-digit column or key part number. <i>s</i> Optional spaces. <i>type</i> Any valid SQL data type except CHAR VARYING, CHARACTER VARYING, NCHAR VARYING, or BIT VARYING (use VARCHAR instead).
<i>DAn</i>	Deletes association <i>n</i> .

CONVERT.SQL Values

Value	Description
<i>Utnsa</i>	Uses another column or association definition. <i>a</i> is the alphabetic synonym designator for the column or association to be used as the preferred definition in the CREATE TABLE statement instead of the definition chosen by CONVERT.SQL.
<i>R[tn]</i>	Redisplays the SQL-formatted definitions for the single column, key part, or association <i>n</i> , or for all columns and associations.
S	Shows the CREATE TABLE statement.
H	Provides help on the options.
M	Provides more help on the options.
Q	Quits CONVERT.SQL and returns to the UniVerse prompt.

CONVERT.SQL Values (Continued)

Creating the Table

After the editing phase, you can create the table by choosing one of the following options from the editing prompt:

Option	Description
X	Executes the CREATE TABLE statement and returns to the UniVerse prompt.
X.SAVEDATA	Saves the file's original data in a file called <i>filename_SQLSAVE</i> , then executes the CREATE TABLE statement and returns to the UniVerse prompt.

Options for Creating Tables

The CREATE TABLE statement includes only the preferred column definitions and the nonoverlapping association definitions.

If an error occurs while creating the table, the process aborts with an appropriate error message. You should then reexecute CONVERT.SQL interactively, correct the problem by editing the SQLDEF file, and try again to create the table.

Example

Here is the dictionary of a simple file called FILEA (using A-descriptors):

Field.....		Type &	Field.....	Output
Name.....	Number	Definition.....	Format	SQL DATA
TYPE				
@ID	D	0		10L
RECID	A	0		10R
FIELD2	A	2		10L
DOUBLE	A	99 F;0;"2";*		10R

The following command converts FILEA into a table, generating a new SQLDEF file (overwriting any existing file named FILEA_SQLDEF):

```
>CONVERT.SQL FILEA CREATE GEN
Analyzing 'FILEA' for conversion to SQL                      18 JUN
1997 17:51
Generating file 'FILEA_SQLDEF' .....
Table name: "FILEA"                                           (SQLDEF was generated 18 JUN
1997 17:51)
Columns:
  00 "RECID" INT FMT '10R'
  00B "@ID" CHAR(10) COL.HDG 'FILEA' FMT '10L'
  01 "SQL_C01" VARCHAR MULTIVALUED
  02 "FIELD2" CHAR(10) FMT '10L'
Preparing to create table .....
Creating Table "FILEA"
Adding Column "RECID"
Adding Column "SQL_C01"
Adding Column "FIELD2"
```

The column definitions shown illustrate that:

- RECID is the preferred definition of column 0.
- A new column SQL_C01 is created for field 1.
- The field DOUBLE is not part of the table definition since it is a correlative.

The updated dictionary display shows that SQL data types have been added for the preferred column names and that several phrases have been inserted:

Field.....		Type &	Field.....	Output
Name.....	Number	Definition.....	Format	SQL DATA TYPE
@ID	D	0		10L
RECID	A	0		10R INTEGER
FIELD2	A	2		10L CHARACTER,10

DOUBLE	A	99 F;0;"2";*	10R	
SQL_C01	D	1	10T	VARCHAR,254
@KEY	PH	RECID		
@SELECT	PH	RECID FIELD2		
@	PH	ID.SUP RECID SQL_C01 FIELD2		
@REVISE	PH	SQL_C01 FIELD2		

The next command restores table FILEA back into a file. Its dictionary is restored to its original form, but if there were data changes they are left intact.

```
>CONVERT.SQL FILEA RESTORE
Restoring Table FILEA to a file.
  Restoring DICT 'FILEA' (using FILEA_SQLDEF)
File restored
```

For more CONVERT.SQL examples, see *UniVerse SQL Administration for DBAs*.

CONVERT.VOC

Use CONVERT.VOC to convert items in a Pick Master Dictionary or records in a Prime INFORMATION VOC file to UniVerse VOC file entries.

Syntax

CONVERT.VOC

Description

When you invoke CONVERT.VOC, you are prompted to select the name of the computer or operating system from which the original Master Dictionary or VOC file came:

Computer	Operating System
ADDS	Mentor
IBM PC-XT	Pick
Microdata	Reality
Prime	PRIMOS
Ultimate	Ultimate
IN2	IN-Pick
Operating Systems	

CONVERT.VOC first converts Pick file definition items (with D/CODE = D) to UniVerse file definition entries (with F1 = F). Q-pointers are implemented as they are in Pick. Proc entries are copied without conversion. If they contain any non-UniVerse verbs or keywords, you must remove them or replace them with UniVerse equivalents.

Next, CONVERT.VOC passes all entries to the DC (dictionary conversion) utility for conversion, then writes them to the VOC file. CONVERT.VOC does not convert entries that do not match the UniVerse template for the particular Pick system from which the Master Dictionary came. Such entries are listed in an error report.

CONVERT.VOC does not overwrite any existing records in the VOC file. Nor does it change or delete the original Master Dictionary or VOC file.

For more information about converting a Pick Master Dictionary, see *UniVerse Guide for Pick Users*.

COPY

Use COPY to copy records to a file, a printer, or the terminal. COPY copies records to other records in the same file and to records in other files. You can also use COPY to rename records.

Syntax

COPY FROM [DICT] *source.file* [**TO** [DICT] *target.file*] [*rec1* [, *new.rec1*] [*rec2* [, *new.rec2*]] ... | **ALL**] [*options*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Copies records to or from the file dictionary.
<i>source.file</i>	The name of the file containing the records to copy. If you do not specify DICT, records are copied from the data file.
<i>target.file</i>	The name of the file to which the records are copied. If you do not specify DICT, records are copied to the data file.
<i>rec</i>	The record ID of the record to be copied. Separate multiple record IDs with spaces.
<i>new.rec</i>	The record ID of the new copy of the record. If you do not specify <i>new.rec</i> , the new record has the same record ID as the old record. If you copy a record to another record in the same file, you must specify <i>new.rec</i> .
ALL	Copies all records in the source file to the target file. The ALL keyword overrides an active select list.

COPY Parameters

options can be any of the following:

Option	Description
CRT	Copies records to the screen.
DELETING	Removes the records from the source file. If a record cannot be copied to the target file, the source file record is not deleted.
FIRST <i>n</i>	Copies the first <i>n</i> records in <i>source.file</i> . COPY creates select list 0 (unless a select list is active) and copies the records specified by the first <i>n</i> record IDs in the select list.
HEX	Lists records in hexadecimal format, regardless of the NLS mode. Use this option with the CRT or LPTR option. You cannot use HEX with the UNICODE option.
ID.SUP	Suppresses list of record IDs on the printer or at the terminal.
LPTR [<i>n</i>]	Sends output to a printer via logical print channel <i>n</i> . If you do not specify <i>n</i> , logical print channel 0 is used. If you specify LPTR and CRT, COPY ignores CRT and sends output to the printer.
NEW.PAGE	Lists each record on a separate page. Use this option when you copy records to the printer or the terminal.
NO.PAGE	Suppresses automatic paging. Use this option when you copy records to the printer or the terminal.
NUM.SUP	Suppresses line numbers. Use this option when you copy records to the printer or the terminal.
OVERWRITING	Overwrites records in the target file that have the same record ID as records being copied from the source file. If you do not specify OVERWRITING, the records in the source file that have the same record ID as records in the target file are not copied.
REPORTING	Lists the status of the OVERWRITE, UPDATING, and DELETING options, and the record IDs of records being copied and the new record IDs.

COPY Options

Option	Description
SQUAWK	Same as REPORTING.
UNICODE	In NLS mode, lists records with each character represented by its 4-digit hexadecimal Unicode value. Use this option with the CRT or LPTR option. You cannot use UNICODE with the HEX option.
UPDATING	Copies a record only if a record with the same record ID exists in <i>target.file</i> . The new record overwrites the old.

COPY Options (Continued)

Description

You can copy records individually, in groups, or all at once. You can also copy records within the source file by using different target record IDs.

To copy records from one file to another, either in groups or individually, list the record IDs on the command line like this:

```
rec1 rec2 rec3 ...
```

To copy records from one file to another, changing the record IDs, separate the original record ID from the new record ID with a comma, like this:

```
rec1,new.rec1 rec2,new.rec2 rec3,new.rec3 ...
```

You can also copy several records and rename only some of them, like this:

```
rec1 rec2,new.rec2 rec3 rec4,new.rec4 ...
```

After you copy compiled I-descriptors from a dictionary, use the COMPILE.DICT command. This avoids any problems that may occur because of changed dependencies or relationships.

COPY can use an active select list 0 if you do not specify record IDs. You cannot use a select list when copying records to the same file, because you cannot change a record's ID using a select list.



Warning: *If NLS is enabled, do not try to copy records between files with different maps. If the source file contains characters that cannot be mapped in the target file, the COPY operation aborts.*

Examples

This example copies the records BOLTS and NUTS from the STOCK.HISTORY file to the STOCK.BACKUP file and displays verbose system messages explaining what it is doing:

```
>COPY FROM STOCK.HISTORY TO STOCK.BACKUP BOLTS NUTS REPORTING
Source file name      = STOCK.HISTORY.
Destination file name = STOCK.BACKUP.
OVERWRITING option    = FALSE.
UPDATING option       = FALSE.
DELETING option        = FALSE.
```

```
"BOLTS" copied to "BOLTS".
"NUTS" copied to "NUTS".
```

```
2 records copied.
```

```
>
```

The next example copies all dictionary entries from the PAYABLES file dictionary to the dictionary of the OLD.VENDORS file:

```
>COPY FROM DICT PAYABLES TO DICT OLD.VENDORS ALL
```

```
25 records copied.
```

The next example selects orders dated September 1992 from the ORDERS file, then copies them from the ORDERS file to the ORDERS.SEPT file:

```
>SELECT ORDERS WITH DATE GE "1 SEPT 92" AND LT "1 OCT 92"
```

```
95 record(s) selected to SELECT list #0.
```

```
>>COPY FROM ORDERS TO ORDERS.SEPT
```

```
You have an active SELECT list.
```

```
Do you wish to copy the records previously SELECTed?
```

```
The first record ID = "8864".
```

```
Enter Y or N: Y
```

```
95 records copied.
```

```
>
```

The next example copies the record DAT from the STOCK.ARCH file to the record DAT_H in the STOCK.H file and deletes the original record from the STOCK.ARCH file:

```
>COPY FROM STOCK.ARCH TO STOCK.H DAT,DAT_H DELETING REPORTING
Source file name      = STOCK.ARCH.
Destination file name = STOCK.H.
OVERWRITING option    = FALSE.
UPDATING option       = FALSE.
DELETING option        = TRUE.
```

```
"DAT" deleted from "STOCK.ARCH".  
"DAT" copied to "DAT_H".
```

```
1 record copied.  
>
```

The next example creates a select list of record IDs that meet a specified criterion, in this case, the value BLUE in the COLOR field. You are asked if you want to copy the selected records. When you answer **Y**, the records are copied.

```
>SELECT CARS WITH COLOR = "BLUE"
```

```
3 record(s) selected to SELECT list #0.
```

```
>>COPY FROM CARS TO CARS.BLUE REPORTING
```

```
Source file name      = CARS.
```

```
Destination file name = CARS.BLUE.
```

```
OVERWRITING option    = FALSE.
```

```
UPDATING option       = FALSE.
```

```
DELETING option       = FALSE.
```

```
You have an active SELECT list.
```

```
Do you wish to copy the records previously SELECTed?
```

```
The first record ID = "HATCHBACK".
```

```
Enter Y or N: Y
```

```
"HATCHBACK" copied to "HATCHBACK".
```

```
"COUPE" copied to "COUPE".
```

```
"GT" copied to "GT".
```

```
3 records copied.
```

```
>
```

COPY.LIST

Use COPY.LIST to copy a saved list from the [&SAVEDLISTS&](#) file to another file.

Syntax

COPY.LIST [*filename*] [*listname*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>filename</i>	The name of the file where the list will be copied. If you do not specify <i>filename</i> , COPY.LIST prompts you to enter one.
<i>listname</i>	The name of a saved list, which is a record in the &SAVEDLISTS& file. If you do not specify <i>listname</i> , COPY.LIST prompts you to enter one.

COPY.LIST Parameters

Description

COPY.LIST is a proc that invokes the COPY processor.

For more information about the [&SAVEDLISTS&](#) file, see Chapter 4, “[Local and System Files.](#)”

Example

This example creates a select list of records from the file SUN.MEMBER, saves the list as the record 1990 in the &SAVEDLISTS& file, then copies the saved list to the VOC file. The record ID of the copied saved list in the VOC file is also 1990.

```
>SELECT SUN.MEMBER WITH YR.JOIN EQ "1990"  
2 record(s) selected to SELECT list #0.  
>>SAVE.LIST 1990  
2 record(s) SAVED to SELECT list "1990".  
>COPY.LIST  
ENTER DESTINATION FILE=VOC  
ENTER LIST ID=1990  
1 record copied.  
>
```



CORE

Use CORE to display statistics about your current memory usage. The statistics include the number of blocks, number of bytes, and average block size for free, used, and total memory.

***Note:** This command is not supported on Windows platforms.*

Syntax

CORE

Example

```
>CORE
Current Memory usage: 46880 bytes (45K)

Free :    7 blocks ( 6%),    5856 bytes (12%), 836.57 average block
Used :    6 blocks (93%),   41024 bytes (87%), 387.01 average block

Total: 113 blocks           46880 bytes           414.86 average block
```

COUNT

Use COUNT to count the number of records in a file.

Syntax

COUNT [DICT | USING [DICT] *dictname*] *filename* [FROM *n*] [*selection*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Counts records in the file dictionary of <i>filename</i> . If you do not specify DICT, records in the data file are counted.
USING [DICT] <i>dictname</i>	If DICT is not specified, uses <i>dictname</i> as the dictionary of <i>filename</i> . If DICT is specified, the dictionary of <i>dictname</i> is used as the dictionary of <i>filename</i> .
<i>filename</i>	Specifies the file containing the records you want counted. You can put <i>filename</i> anywhere in the sentence. COUNT uses the first word in the sentence that has a file descriptor in the VOC file as the filename.
FROM <i>n</i>	Specifies that select list <i>n</i> is to be used.
<i>selection</i>	A selection expression specifying the conditions that data in a record must meet for the record to be counted. A selection expression begins with the keyword WITH. For syntax details, see the WITH keyword.

COUNT Parameters

Description

If you enter COUNT with no options, it counts and reports the total number of records in the file.

Example

This example counts all records in the file SUN.MEMBER with LNAME equal to BROWN:

```
>COUNT SUN.MEMBER WITH LNAME EQ BROWN
```

```
1 records counted.
```

CP

Use CP to print records on the system printer. CP prints only to print channel 0.

Syntax

CP [DICT] *filename* [*records* | *] [*options*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
<i>filename</i>	The file containing the record you want to print.
<i>records</i>	The record IDs of records you want to print.
*	Specifies all records in <i>filename</i> .

CP Parameters

options can be any of the following:

Option	Description
–FORM.FEED	Lists each record on a separate page.
–HEX	Copies data in hexadecimal format. You can use this option whether NLS mode is on or off. You cannot use –HEX with the UNICOD option.
–NO.PAGE	Suppresses automatic paging.
–UNICODE	In NLS mode, lists records with each character represented by its 4-digit hexadecimal Unicode value. You cannot use –UNICODE with the –HEX option.

CP Options

Example

The following example prints the record LIST in the VOC file:

```
>CP VOC LIST
```

CREATE.ENCRYPTION.KEY

Use the CREATE.ENCRYPTION.KEY command to create an encryption key in the UniVerse key store. We recommend that you create a password for the key.

Syntax

CREATE.ENCRYPTION.KEY *key.id* [*password*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>key.id</i>	The encryption key ID.
<i>password</i>	The password for <i>key.id</i> .

CREATE.ENCRYPTION.KEY Parameters

Note: We suggest that the password you create is a phrase that is hard to guess, but easy to remember, using a combination of ASCII characters and digits. If a password contains a space (“ ”), you must use quotation marks to enclose the password.



CREATE.ENCRYPTION.WALLET

Use the CREATE.ENCRYPTION.WALLET command to create an encryption wallet, which contains encryption keys and passwords, in the UniVerse key store.

Syntax

CREATE.ENCRYPTION.WALLET <*wallet_ID*> <*wallet_password*>

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>wallet_id</i>	The encryption wallet ID.
<i>wallet_password</i>	The password for the encryption wallet.

CREATE.ENCRYPTION.KEY Parameters

CREATE.FILE

Use CREATE.FILE to create a UniVerse file. CREATE.FILE creates the data file, the file dictionary, and the file definition record in the VOC file.

Syntax

```
CREATE.FILE [ DICT | DATA ] [ filename [ ,datafile ] ] [ type ] [ modulo ]  
[ separation ] [ parameter [ value ] ] ... [ 64BIT | 32BIT ] [ description ]
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies only the file dictionary. If you do not supply file specifications on the command line, the following default specifications are used for a file dictionary: type3 modulo1 separation2
DATA	Specifies only the data file.
<i>filename</i>	The UniVerse filename. CREATE.FILE creates the data file with the name <i>filename</i> and the file dictionary with the name D_ <i>filename</i> . If you do not specify <i>filename</i> , CREATE.FILE prompts for it. <i>filename</i> must follow the UniVerse naming conventions. For more information, see “File Naming Conventions” on page 161.
<i>datafile</i>	The name of a data file, used when creating dictionaries of files with multiple data files. Use a comma (no spaces allowed) to separate the data file name from the filename of the UniVerse file.

CREATE.FILE Parameters

Parameter	Description
<i>type</i>	The Universe file type for the file you want to create. Type 1 or type 19 files are not hashed and are usually used to store text files such as BASIC programs. Types 2 through 18 are hashed files. Type 25 is a balanced tree file. Type 30 is a dynamic file. You can also specify the keyword DYNAMIC for a type 30 file. If you do not specify <i>type</i> , CREATE.FILE prompts for it.
<i>modulo</i>	An integer from 1 through 8,388,608 defining the number of groups in the file. If you do not specify <i>modulo</i> , CREATE.FILE prompts for <i>modulo</i> and <i>separation</i> . <i>modulo</i> is ignored for nonhashed and dynamic files.
<i>separation</i>	An integer from 1 through 8,388,608 specifying the group buffer size in 512-byte blocks. If you do not specify <i>separation</i> and one is needed, the file is created with a separation of 4, unless you are in prompting mode. If you press ENTER in response to the <i>separation</i> prompt, the file is not created. <i>separation</i> is ignored for nonhashed and dynamic files.
<i>description</i>	Additional information about the file. This description is put in field 1 of the VOC file entry starting in the third character position. A file is a file descriptor type F. You can display the description as a help message for <i>filename</i> by entering ?filename . You must specify <i>description</i> in quotation marks when you create dynamic files.

CREATE.FILE Parameters (Continued)

parameter specifies a dynamic file parameter. *parameter* can be any of the following:

Parameter	Description
GENERAL	Specifies that the general hashing algorithm should be used for the dynamic file. GENERAL is the default.
SEQ.NUM	Specifies that a hashing algorithm suitable for sequential numbers should be used for the dynamic file. You should use this hashing algorithm only for records with IDs that are mainly numeric, sequential, and consecutive.
GROUP.SIZE { 1 2 }	Specifies the size of each group in the file. The argument 1 specifies a group size of 2048 bytes, which is equivalent to a separation of 4. The argument 2 specifies a group size of 4096 bytes, which is equivalent to a separation of 8. A group size of 1 is the default.

Dynamic File Parameters

Parameter	Description
MINIMUM.MODULUS <i>n</i>	Specifies the minimum modulo of the file. This keyword takes an integer argument greater than 1. This value is also the initial value of the modulo of the dynamic file. A minimum modulo of 1 is the default.
SPLIT.LOAD <i>n</i>	Specifies the level at which the file's modulo is increased by 1. SPLIT.LOAD takes a numeric argument indicating a percentage of the space allocated for the file. When the data in the file exceeds the space allocated for the file, the data in one of the groups divides equally between itself and a new group, to increase the modulo by 1. The default SPLIT.LOAD is 80%.
MERGE.LOAD <i>n</i>	Specifies the level at which the file's modulo is decreased by 1. MERGE.LOAD takes a numeric argument indicating a percentage of the space allocated for the file. When the data in the file uses less than the space allocated for the file, the data in the last group of the file merges with another group, to decrease the modulo by 1. The default MERGE.LOAD is 50%.

Dynamic File Parameters (Continued)

Parameter	Description
LARGE.RECORD <i>n</i>	Specifies the size of a record to be considered too large to be included in the primary group buffer. This keyword takes an argument that can be specified as an integer or a percentage. Specified as an integer, the argument is the number of bytes a record must contain to be considered a large record. Specified as a percentage, the argument is a percentage of the group size. When the size of a record exceeds the value specified, the data for the record is placed in an overflow buffer, but the record ID is place in the primary buffer. This method of large record storage increases access speed. The default LARGE.RECORD size is 80%.
RECORD.SIZE <i>n</i>	Causes the values for group size, and large record size to be calculated based on the value of the estimated average record size specified. RECORD.SIZE takes an argument of your estimate of the average record size for the dynamic file, specified in bytes. RECORD.SIZE does not limit the size of records. If you specify a value for group size, or large record size, the value you specify overrides the value calculated by RECORD.SIZE.
MINIMIZE.SPACE	Calculates the values for the split load, merge load, and the large record size to optimize the amount of space required by the file at the expense of access time. If you specify a value for split load, merge load, or large record size, the value you specify overrides the value that is calculated by MINIMIZE.SPACE. If MINIMIZE.SPACE and RECORD.SIZE are specified, the value for large record size calculated by MINIMIZE.SPACE overrides the value calculated by RECORD.SIZE.

Dynamic File Parameters (Continued)

The following options override the current setting of the 64BIT_FILES configurable parameter:

Option	Description
64BIT	Creates a 64-bit file on a UniVerse system using 32-bit file systems.
32BIT	Creates a 32-bit file on a UniVerse system using 64-bit file systems.

64-bit File Options

Description

CREATE.FILE creates or modifies the file definition in the VOC file. In addition, CREATE.FILE allocates space for the data file and the file dictionary and creates a default field definition in the dictionary for record IDs.

Any existing UniVerse file can be converted into a file made up of multiple data files by issuing the following command:

```
CREATE.FILE DATA filename,datafile type modulo separation
```

If a file already comprises multiple data files, the following command expands *filename* to *filename.filename*:

```
CREATE.FILE DATA filename
```

Dynamic files exist to make file management easier for users. The default parameters are set so most dynamic files work correctly. For some files, you can increase efficiency by changing the default file parameters using CONFIGURE.FILE. You should verify that your changes to the default parameters actually increase the efficiency of the file using the [ANALYZE.FILE](#) command.

If NLS is enabled, the CREATE.FILE command sets a map name for the file using the value set by the NLSNEWFILEMAP or NLSNEWDIRMAP configurable parameters. For more information, see the *UniVerse NLS Guide*.

File Naming Conventions

UniVerse filenames can be up to 255 characters long, but it makes sense to keep them as short as you can. If your operating system has a 14-character limit on file names, UniVerse truncates the file name to 9 characters and adds a 3-digit sequencer. On these operating systems it is important to keep the first 9 characters of each file name unique.



Note: *On Windows platforms, UniVerse is designed to run in the Windows file system (NTFS). But Windows platforms also support the MS-DOS FAT file system which limits file names to 8 characters with a 3-character extension, and has a further set of characters that are not permitted in file names. UniVerse makes no special provision for these file names or special characters. If you want to store UniVerse files in an MS-DOS FAT file system, you must follow the MS-DOS conventions when you name UniVerse files.*

File names in UniVerse can contain any character except CHAR(0). If you use spaces and control characters in your file names, you must enclose the entire file name in quotation marks whenever you use it in a UniVerse command, including the CREATE.FILE command.

Operating systems limit the characters that can be used in a file name. You can still specify these characters in UniVerse file names, but UniVerse maps the problem characters at the operating system level. This means that some UniVerse file names may look different when viewed from the operating system. This does not affect the file name you use on the command line. The mapping UniVerse uses depends on the operating system.

UNIX:

This character...	Maps to...
/	?\
?	??
empty filename	?0
. (leading period)	?.

Mapping for UNIX

Windows Platforms:

This character...	Maps to...
/	%S
?	%Q
empty filename	%
"	%D
%	%%
*	%A
:	%C
<	%L
(vertical bar)	%V
>	%G
\	%B
↑ (up-arrow)	↑↑ (up-arrow)
ASCII 1 through ASCII 26	↑A through ↑Z
ASCII 27 through ASCII 31	↑1 through ↑5

Mapping for Windows NT

Examples

This example creates the data file and the dictionary of a UniVerse file called OVERDUE. The data file is a type 2 file with modulo and separation of 1. Field 1 of the file descriptor contains the description “Overdue accounts receivable.” The dictionary uses the default file parameters: file type 3, modulo 1, and separation 2.

```
>CREATE.FILE OVERDUE 2 1 1 Overdue accounts receivable
Creating file "OVERDUE" as Type 2, Modulo 1, Separation 1.
Creating file "D_OVERDUE" as Type 3, Modulo 1, Separation 2.
Added "@ID", the default record for Retrieve, to "D_OVERDUE".
```

The next example creates a file dictionary for the file LONG.OVERDUE. The file type is 4, the modulo is 3, and the separation is 2.

```
>CREATE.FILE DICT LONG.OVERDUE 4 3 2
Creating file "D_LONG.OVERDUE" as Type 4, Modulo 3, Separation 2.
Added "@ID", the default record for Retrieve, to "D_LONG.OVERDUE".
```

The next example creates a new data file for the UniVerse file LONG.OVERDUE. The new data file is called MONTH. It is a type 2 file with a modulo of 3 and a separation of 2.

```
>CREATE.FILE DATA LONG.OVERDUE,MONTH 2 3 2
Creating a multilevel data file.
Creating file "LONG.OVERDUE/MONTH" as Type 2, Modulo 3, Separation 2.
```

The following table shows the VOC entries for the files created by the previous examples:

Command	VOC Entry
CREATE.FILE OVERDUE 2 1 1	OVERDUE 001 F 002 OVERDUE 003 D_OVERDUE
CREATE.FILE DICT LONG.OVERDUE 4 3 2	LONG.OVERDUE 001 F 002 003 D_LONG.OVERDUE
CREATE.FILE DATA LONG.OVERDUE,MONTH 2 3 2	LONG.OVERDUE 001 F 002 LONG.OVERDUE 003 D_LONG.OVERDUE 004 M 005 006 007 MONTH 008 MONTH
VOC Entries	

CREATE.INDEX

Use CREATE.INDEX to create a secondary index for a file.

Syntax

CREATE.INDEX [DICT | USING [DICT] *dictname*] [*filename*]
[AT *account*] [*fields*] [NO.NULLS] [COLLATE { *locale* | OFF }]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies a file dictionary.
USING <i>dictname</i>	If DICT is not specified, uses <i>dictname</i> as the dictionary of <i>filename</i> . If DICT is specified, the dictionary of <i>dictname</i> is used as the dictionary of <i>filename</i> .
<i>filename</i>	The name of the file containing fields you want to index. If you do not specify <i>filename</i> , CREATE.INDEX prompts you for it.
<i>account</i>	Specifies the UniVerse account where you want to store the indexes you create. <i>account</i> must be defined in the UV.ACCOUNT file. If <i>filename</i> already has indexes, you cannot use the AT clause because all indexes are stored as files in one account directory.

CREATE.INDEX Parameters

Parameter	Description
<i>fields</i>	A list of fields separated by spaces to use as secondary access keys. If <i>fields</i> is not specified, CREATE.INDEX prompts you to enter a single field.
NO.NULLS	Specifies not to index empty secondary key values in the newly created indexes. You can save disk space and processing time using this facility, but reports generated from empty-string-suppressed secondary keys do not contain values for the empty keys.
COLLATE	If NLS is enabled, specifies the name of the locale whose Collate convention you want to associate with an index. OFF specifies not to associate specific collate information with the index; in this case the index uses the default sort order. If you do not specify COLLATE, UniVerse uses the Collate convention of the current locale on the index.

CREATE.INDEX Parameters (Continued)

Description

Setting up a secondary index is a two-step process. First you create the index with CREATE.INDEX. Then you build the index with [BUILD.INDEX](#). You must use the BUILD.INDEX command to build the initial index, even if the file you want to index contains no records. If the indexed file contains records when you create the index, a warning reminds you to build the index.

The file is not locked for use by other users while the index is created.

If CREATE.INDEX cannot find a field definition in the file dictionary, it looks for it in the VOC file.

After an index is created for a field, UniVerse no longer consults the file dictionary for information about the field. The index stores information such as a field number or I-type expression. This ensures the accuracy of the index even if the dictionary changes.

For a field to be indexed as a multivalued field, it must be defined as a multivalued field in the file dictionary. Singlevalued fields are indexed as single keys without regard to value marks.



I-type expressions that change based on data external to the environment, such as the date or the user ID, are not suitable for use as secondary keys.

If you change an I-type expression in the file dictionary, you must also change the index. To change the index, use [DELETE.INDEX](#) to delete the old index, create a new index with [CREATE.INDEX](#), and build the new index with [BUILD.INDEX](#).

***Note:** When you create indexes, be careful when using the [TRANS](#) or [XLATE](#) functions, or the [Tfile](#) conversion code to translate data back to the main data file. While indexes are updated, the main data file retains a write lock that blocks read access to the group containing the written record. When the translate key is [@ID](#), [TRANS](#), [XLATE](#), and [Tfile](#) are optimized to read the written record from memory to bypass the write lock. Any other attempt to translate to a record in the locked group results in a deadlock. Also note that an I-type subroutine containing an [OPEN](#) and [READ](#) statement that tries to reread the record from the main data file results in a deadlock.*

NLS Mode

[CREATE.INDEX](#) associates collating information for the current locale with the index unless you explicitly override this with the [COLLATE](#) option. For example, if you use the [SET.LOCALE OFF](#) command to disable, you can override this by using [CREATE.INDEX](#) with the [COLLATE](#) option.

Examples

This example creates an index for the [MONTHLY.STATEMENT.FLAG](#) field of the [CREDITORS](#) file. Empty strings are not included in the index.

```
>CREATE.INDEX CREDITORS MONTHLY.STATEMENT.FLAG NO.NULLS  
>
```

The next example creates two indexes, one on the [BALANCE](#) field and one on the [CREDIT.RATING](#) field of the [DEBTORS](#) file:

```
>CREATE.INDEX DEBTORS BALANCE CREDIT.RATING  
>
```

The next example creates an index on the BALANCE field of the DEBTORS file. The field definitions for this file are stored in the dictionary of the CUSTOMERS file, and the index is stored in the INDICES file.

```
>CREATE INDEX DEBTORS BALANCE USING DICT CUSTOMERS AT INDICES  
>
```

The next example creates an index on the LAST.NAME field, specifying that it be sorted using French locale conventions:

```
>CREATE INDEX CUSTOMERS LAST.NAME COLLATE FR-FRENCH  
>
```

CREATE.LDIR

Use CREATE.LDIR to create the log directory for the transaction logging system. You must be a UniVerse Administrator logged in to the UV account to use CREATE.LDIR, and transaction logging must be disabled or inactive.

Syntax

CREATE.LDIR *pathname*

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>pathname</i>	The absolute path of the log directory.

CREATE.LDIR Parameter

Description

The log directory contains the log files where warmstart and data transactions are logged. Before you can use transaction logging, you must create a log directory and a set of log files to store logged updates to recoverable files. If the log files in the log directory fill up and you need more space, you can create another log directory for them on another partition.

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

The path of the log file is stored in the LOGS.DIR record in the dictionary of the UV_LOGS file.

If transaction logging is currently enabled, use [SHUTDOWN.RECOVERY](#) to disable it before using CREATE.LDIR.

Example

This example creates the log directory on the */u2* partition:

```
>CREATE.LDIR /u2/translog
```

CREATE.LFILE

Use CREATE.LFILE to create log files in the transaction logging system log directory. You must be a UniVerse Administrator logged in to the UV account to use CREATE.LFILE.

Syntax

CREATE.LFILE *size files*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>size</i>	The size of the log files, in bytes.
<i>files</i>	The number of log files you want to create.

CREATE.LFILE Parameters

Description

CREATE.LFILE creates log files with names in the following format:

lgnnn

Log files are numbered in sequence, starting with the number stored in the LOG.NEXT record in the dictionary of the UV_LOGS file.

For each log file, a corresponding record is created in the UV_LOGS file.

The size of the log files you create depends on the volume of updates you expect. The block size of log files is 512.

Example

This example creates nine log files in the log directory. The size of each file is 500K.

```
>CREATE.LFILE 512000 9
```

```
Creating lg348 (Each '*' = 1024 bytes.)
```

```
*****  
*****  
*****  
*****  
*****  
*****  
*****lg348 Created.
```

```
Creating lg349 (Each '*' = 1024 bytes.)
```

```
*****  
*****  
*****  
*****  
*****  
*****  
*****lg349 Created.
```

```
.  
.  
.
```

CS

Use CS to clear the screen and return the cursor to the home position. CS is a synonym for the [CLR](#) command.

Syntax

CS



CSH

Use CSH to invoke a UNIX C shell (*csh*) from within UniVerse.

Note: *This command is not supported on Windows platforms.*

Syntax

CSH [`-c "command"` | *script*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>command</i>	The command text. It must be preceded by the flag <code>-c</code> . To avoid confusion, enclose <i>command</i> in quotation marks.
<i>script</i>	The pathname of a UNIX shell script to be executed. The shell script file does not need execute privileges.

CSH Parameters

Description

Once you invoke a C shell, you can execute any UNIX command. You can also give CSH an argument to execute a single command or a shell script without exiting UniVerse. To return to the UniVerse prompt, enter **exit** or press Ctrl-D.

Availability of the C shell is system-dependent. See your UNIX documentation for a discussion of C shell features.

Use the [SH](#) command to execute UNIX Bourne shell (*sh*) commands.

Example

This example invokes the C shell and runs the UNIX command `ls` to list the contents of the current directory:

```
>CSH -c"ls -lt"
total 707
drwxrwxrwx  2 susan  acctg          512 Jul  6 11:43 I_CREDITORS
-rw-rw-r--  1 susan  acctg         1536 Jul  6 11:43 CREDITORS
-rw-rw-r--  1 susan  acctg         2048 Jul  6 11:42 D_CREDITORS
-rw-r--r--  1 susan  acctg        49152 Jul  6 11:41 VOC
drwxrwxrwx  2 susan  acctg          512 Jul  2 16:55 I_ORDERS
-rw-r--r--  1 susan  acctg        3072 Jul  2 16:55 D_ORDERS
drwxrwxr-x  2 susan  acctg          512 Jul  2 16:43 LONG.OVERDUE
-rw-rw-r--  1 susan  acctg        4096 Jul  2 16:41 D_LONG.OVERDUE
-rw-rw-r--  1 susan  acctg        2048 Jul  2 16:40 D_OVERDUE
-rw-rw-r--  1 susan  acctg        1536 Jul  2 16:40 OVERDUE
```

CT

Use CT to display records on the terminal.

Syntax

CT [DICT]*filename* [*records* | *] [*options*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
<i>filename</i>	The file containing the record you want to display.
<i>records</i>	The record IDs of records you want to display.
*	Specifies all records in the file.

CT Parameters

options can be any of the following:

Option	Description
–FORM.FEED	Lists each record on a separate page.
–HEX	Copies data in hexadecimal format. You can use this option whether NLS mode is on or off. You cannot use –HEX with the UNICOD option.
–NO.PAGE	Suppresses automatic paging.
–UNICODE	In NLS mode, lists records with each character represented by its 4-digit hexadecimal Unicode value. You cannot use –UNICODE with the –HEX option.

CT Options

Example

This example displays the record LIST in the VOC file:

```
>CT VOC LIST
      LIST
0001 V
0002 LIST
0003 Q
0004 S
```

DATA

Use DATA in a paragraph to specify a response to an input request. To use the DATA statement, enter the program name or verb name on one line, then enter the DATA statement on the next lines.

Syntax

DATA *data*

Parameter

The following table describes the parameter for the syntax.

Parameter	Description
<i>data</i>	The response to the input request from a UniVerse BASIC program or statement, or to a verb called by a paragraph.

DATA Parameter

Description

When a BASIC program, sentence, paragraph, or command executed in the paragraph requires input, it uses the data from the first DATA statement instead of prompting for input from the terminal. The second time input is required, the data from the second DATA statement is used, and so on. When the data from all previously executed DATA statements has been used, a program, sentence, paragraph, or command that requires input prompts for a response from the terminal.

You can use inline prompting with a DATA statement.

You can use DATA statements only in a paragraph. They have no function on the command line.

Example

This example represents a paragraph as it appears in a VOC file:

```
001 PA
002 RUN BP DUE . LIST
003 DATA <<ENTER DATE>>
004 DATA YES
005 RUN BP VENDORS
006 DATA <<ENTER MONTH>>
```

DATE

Use DATE to display the current day, date, and time on your terminal.

Syntax

DATE

Description

DATE uses a 12-hour clock to display the time.

You can change the format of the date from the standard United States format (month, day, year) to any format (for example, day, month, year) using the [DATE.FORMAT](#) command.

If NLS is enabled, the Time convention of the current locale formats the date. To override the Time convention, use the DATE.FORMAT command. For more information, see the *UniVerse NLS Guide*.

Example

```
>DATE
Thursday, July 06, 1995 11:57am
>
```

DATE.FORMAT

Use DATE.FORMAT to set or change the default date format.

Syntax

DATE.FORMAT [ON | OFF | *conversion* | INFORM]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
ON	Specifies international date format. This is the default format if you specify DATE.FORMAT without options. You cannot use this option if NLS is enabled.
OFF	Specifies United States date format. You cannot use this option if NLS is enabled.
<i>conversion</i>	Any valid date conversion code.
INFORM	Sets the system variable @SYSTEM.RETURN.CODE to the value of the DATE.FORMAT flag (0 = US, 1 = European).

DATE.FORMAT Parameters

Description

In international format, the day of the month appears first. In United States format, the month appears first.

Note: If NLS is enabled, you cannot turn the date format on or off; you can use DATE.FORMAT only to specify a date conversion.



If you specify a date conversion, the conversion format becomes the default date format. If NLS is enabled, the conversion format overrides the Time convention of the current locale setting (for information about locales, see the *UniVerse NLS Guide*). Date conversions specified in file dictionaries or in ICONV or OCONV functions use the default date format except where they specifically override it.

If, after you specify a date conversion, you want to reset the date format to the system default (which may be the current NLS locale convention), use the following command:

```
>DATE.FORMAT D
```

DATE.FORMAT changes only the format. It does not change the internal representation of dates as returned by the DATE function or the @DATE system variable.

DATE.FORMAT does not display the date. Use the [DATE](#) command to display the date.

The date format you specify remains in effect until you log out.

Examples

```
>DATE.FORMAT ON
>DATE
Wednesday, 10 May 1995 09:35am
>DATE.FORMAT OFF
>DATE
Wednesday, May 10, 1995 09:35am
```

This example shows how you can use a date conversion to change the default date format. The dictionary entry defining the DATE field in the ORDERS file specifies a date conversion of D2:

```
DATE
0001 D
0002 1
0003 D2
0004 Order Date
0005 9R
0006 S
```

The date conversion in the following example specifies the order *year-month-day*. The format modifier in brackets specifies that two characters be used to display the month, without suppressing leading zeros. The second DATE.FORMAT command resets the date format to the system default.

```
>DATE.FORMAT D2-YMD[,2]
>LIST ORDERS DATE
LIST ORDERS DATE 03:10:57pm 95-09-14 PAGE 1
@ID.. Order Date

10004      92-08-22
10006      92-04-22
10002      92-07-14
10005      92-11-25
10003      92-03-07
10001      92-02-11
10007      92-07-06

>
>DATE.FORMAT D
>LIST ORDERS DATE
LIST ORDERS DATE 03:13:25pm 14 Sep 1995 PAGE 1
@ID.. Order Date

10004      22 AUG 92
10006      22 APR 92
10002      14 JUL 92
10005      25 NOV 92
10003      07 MAR 92
10001      11 FEB 92
10007      06 JUL 92

7 records listed.
>
```

DEACTIVATE.ENCRIPTION.KEY

Use the DEACTIVATE.ENCRIPTION.KEY command to deactivate one or more encryption keys. This command is useful to deactivate keys to make your system more secure.

Syntax

DEACTIVATE.ENCRIPTION.KEY *key.id password* [ON <*hostname*>]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>key.id</i>	The key ID to deactivate.
<i>password</i>	The password corresponding to <i>key.id</i> .
ON <i>hostname</i>	The name of the remote host on which you want to deactivate the encryption key.

DEACTIVATE.ENCRIPTION.KEY Parameters

Note: You can deactivate only keys with password protection with this command. Keys that do not have password protection are automatically activated and cannot be deactivated.



DEACTLIST

Use DEACTLIST to deactivate files for transaction logging. You must be a UniVerse Administrator logged in to the UV account to use DEACTLIST.

Syntax

DEACTLIST *listname*

Parameter

The following table describes each parameter of the syntax.

Parameter	Description
<i>listname</i>	The record ID of a select list created and saved in the &SAVEDLISTS& file by the MKFILELIST command.

DEACTLIST Parameter

Description

Use the MKFILELIST command to create a list of files that you want to activate and deactivate for transaction logging. Use the ACTLIST command to activate the files in the list.

For information about transaction logging, see *UniVerse Transaction Logging and Recovery*.

Example

This example deactivates all files listed in the saved list INV.FILE.LIST:

```
>DEACTLIST INV.FILE.LIST
```

DECATALOG

Use DECATALOG to delete a locally cataloged program from your VOC file and to delete the object code for that program. DECATALOG does not delete the source code of the program. (Use [DELETE](#) to do this.)

Syntax

DECATALOG [*filename* [*program*]]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>filename</i>	The name of the program file. File names can include only ASCII characters, no multibyte characters. If you do not specify <i>filename</i> , you are prompted for it.
<i>program</i>	The name of the program to be decataloged. Program names can include only ASCII characters, no multibyte characters. If you do not specify <i>program</i> , DECATALOG prompts for it. If there is an active select list, it is used for <i>program</i> .

DECATALOG Parameters

Example

This example deletes the verb entry TEST from the VOC file and deletes the object code record TEST from the file BP.O:

```
>DECATALOG BP TEST
TEST removed from VOC.  Object DELETED from BP.O.
```

DECRYPT.FILE

The DECRYPT.FILE command decrypts data in in a file or in the fields you specify.

Syntax

DECRYPT.FILE {<filename> <type> <modulo> <separation> | <30 | dynamic> parameter [value]...} <USING partition> < {WHOLERECORD | *fieldname*} ,key[,*pass*] [*fieldname*,key[,*pass*]]...>

Most of the DECRYPT.FILE parameters are the same as the RESIZE command parameters. If the file you are decrypting is empty, you do not need to specify any of the RESIZE parameters. If the file you are decrypting is not empty, and you know that the file needs resizing because decrypting the file will change the record size, you should specify the RESIZE parameters.

The following table describes each parameter of the syntax.

Parameter	Description
<i>filename</i>	The UniVerse file name. If you do not specify <i>filename</i> , DECRYPT.FILE prompts for the name. <i>filename</i> must follow the UniVerse naming conventions. For more information about naming conventions, see “File Naming Conventions” in <i>UniVerse User Reference</i> .
<i>type</i>	The UniVerse file type for the file you are decrypting. Type 1 or type 19 files are not hashed and are usually used to store text files such as BASIC programs. Types 2 through 18 are hashed files. Type 25 is a balanced tree file.
<i>modulo</i>	The modulo for the file you are decrypting. The modulo should be an integer from 1 through 8,388,608 defining the number of groups in the file. UniVerse ignores <i>modulo</i> if you specify a nonhashed or dynamic file type.
<i>separation</i>	The separation for the file you are decrypting. The separation should be an integer from 1 through 8,388,608, specifying the group buffer size is 512-byte blocks. UniVerse ignores <i>separation</i> if you specify a nonhashed or dynamic file type.
30	Decrypts a dynamic file.

DECRYPT.FILE Parameters

Parameter	Description
dynamic	Decrypts a dynamic file.
USING <i>partition</i>	Specifies the path of the work area that DECRYPT.FILE will use for creating the necessary temporary files. For example, the following command decrypts SUN.MEMBER as a dynamic file, and creates the temporary files it needs in the partition <i>/u4</i>: >DECRYPT.FILE SUN.MEMBER DYNAMIC USING /u4 DECRYPT.FILE moves the files back into the correct directory after creating the SUN.MEMBER file.
WHOLERECORD	Specifies to fully decrypt every record in the file.
<i>fieldname,key,pass</i>	Specifies the field name to decrypt, and the key, and password to use. You can use a different key for each field. If you do not specify a password, but created the key using password protection, UniVerse prompts for the password. If several fields use the same password, you only have to specify it once, at the first field that uses that key.
<i>fieldname</i>	The name of the field to decrypt.
<i>key</i>	The key ID to use for the field decryption.
<i>pass</i>	The password corresponding to the <i>key</i> .

DECRYPT.FILE Parameters (Continued)

Specify the following parameters only for dynamic files:

Parameter	Description
GENERAL	Specifies the general hashing algorithm for a dynamic file. GENERAL is the default.
SEQ.NUM	Specifies a hashing algorithm suitable for sequential numbers for a dynamic file. Use this hashing algorithm only for records with IDs that are mainly numeric, sequential, and consecutive.
GROUP.SIZE { 1 2 }	Specifies the size of each group in the file, either 1 or 2. 1 specifies a group size of 2048 bytes, which is equivalent to a separation of 4. 2 specifies a group size of 4096 bytes, which is equivalent to a separation of 8. A group size of 2048 (GROUP.SIZE 1) is the default.
MINIMUM.MODULUS <i>n</i>	Specifies the minimum modulo of the file, an integer value greater than 1. This value is also the initial value of the modulo of the dynamic file. A minimum modulo of 1 is the default.
SPLIT.LOAD <i>n</i>	Specifies the level at which the file's modulo is increased by 1. SPLIT.LOAD takes a numeric argument indicating the percentage of space allocated for the file. When the data in the file exceeds the specified percentage of the space allocated for the file, the data in one of the groups is divided equally between itself and a new group, to increase the modulo by 1. The default SPLIT.LOAD is 80%.
MERGE.LOAD <i>n</i>	Specifies the level at which the file's modulo is decreased by 1. MERGE.LOAD takes a numeric argument indicating the percentage of space allocated for the file. When the data in the file is less than the specified percentage of the space allocated for the file, the data in the last group of the file is merged with another group, to decrease the modulo by 1. The default MERGE.LOAD is 50%.

DECRYPT.FILE Parameters for Dynamic Files

Parameter	Description
LARGE.RECORD <i>n</i>	Specifies the size of a record considered too large to be included in the primary group buffer, specified as an integer or a percentage. Specified as an integer, the value is the number of bytes a record must contain to be considered a large record. Specified as a percentage, the value is a percentage of the group size. When the size of a record exceeds the specified value, the data for the record is put in an overflow buffer, but the record ID is put in the primary buffer. This method of large record storage increases access speed. The default LARGE.RECORD size is 80%.
RECORD.SIZE <i>n</i>	Calculates the values for group size and large record size based on the value of the estimated average record size specified. The value is your estimate of the average record size for the dynamic file, specified in bytes. RECORD.SIZE does not limit the size of records. If you specify a value for group size (GROUP.SIZE) or for large record size (LARGE.RECORD), those values override the value calculated by RECORD.SIZE.
MINIMIZE.SPACE	Calculates the best amount of space required by the file (at the expense of access time), using the values for the split load, merge load, and large record size. If you specify values for split load, merge load, or large record size, those values override the value calculated by MINIMIZE.SPACE. If you specify MINIMIZE.SPACE and RECORD.SIZE, the value for large record size calculated by MINIMIZE.SPACE is used above the value calculated by RECORD.SIZE.

DECRYPT.FILE Parameters for Dynamic Files (Continued)

If the encrypted file was created using the WHOLERECORD keyword, you should specify WHOLERECORD when decrypting the file. If the file was not encrypted using the WHOLERECORD keyword, do not specify WHOLERECORD when decrypting the file.

DEFINE.DF

Use DEFINE.DF to create or modify a UniVerse distributed file. You can use distributed files when you want to organize data files logically, by functional groups. You can also use a distributed file when you want to store records in the same file but on different disk partitions, or when a file exceeds the size limit for files on your system.

Syntax

To create a new distributed file or to add new part files to an existing distributed file:

```
DEFINE.DF [ DATA ] dist.filename [ [ ADDING ] part.name [ part# ]  
[ part.name [ part# ] ] ... ] [ algorithm ] [ FORCE ]
```

To remove part files from a distributed file:

```
DEFINE.DF [ DATA ] dist.filename REMOVING part.name [ part.name ] ...  
[ RETAIN ] [ FORCE ]
```

To change the part number of a part file:

```
DEFINE.DF [ DATA ] dist.filename part.name part# [ part.name part# ] ...  
[ FORCE ]
```

To change the partitioning algorithm of a distributed file:

```
DEFINE.DF [ DATA ] dist.filename algorithm [ FORCE ]
```

To cancel the part number and partitioning algorithm of a part file:

```
DEFINE.DF part.name CANCEL
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DATA	Specifies that only the data file is to be created or modified.
<i>dist.filename</i>	The name of the distributed file you are creating or modifying.
ADDING	Adds a new part file to an existing distributed file. If you are creating a new distributed file, the ADDING keyword is optional. If you are adding a new part file to an existing distributed file, you must use the ADDING keyword.
<i>part.name</i>	The name of a UniVerse file you are adding to or removing from a distributed file, or whose part number you are changing. The VOC file and nonhashed files cannot be part files.
<i>part#</i>	A unique part number to be associated with the specified part file. If you are creating a new distributed file or adding a new part file to an existing distributed file, you must specify the part number. If the part file is already part of another distributed file, you need not specify <i>part#</i> .
<i>algorithm</i>	An expression that defines the partitioning algorithm for the distributed file. If you are creating a new distributed file or adding a new part file to an existing distributed file, you must specify a partitioning algorithm. See “Specifying the Partitioning Algorithm” on page 193 for more information.
FORCE	Forces the part number or the algorithm to be changed even if the part file belongs to more than one distributed file.
REMOVING	Removes the specified part file from the distributed file. REMOVING also removes the part number and partitioning algorithm from the part file, unless you specify the RETAIN option.
RETAIN	Used only with the REMOVING clause, RETAIN retains the part number and algorithm in the part file after it is removed from the distributed file.
CANCEL	Cancels the part number and partitioning algorithm of a part file.

DEFINE.DF Parameters

Description

A distributed file is made up of one or more part files. Part files are standard UniVerse hashed files (including dynamic files). Type 1 and type 19 files cannot be part files.

Each part file has a unique part number associated with it. A distributed file uses a partitioning algorithm to distribute records among part files. The algorithm can be user-defined or system-defined. The part number and the partitioning algorithm, or a reference to it, are stored with the part file.

When you create a new distributed file, DEFINE.DF creates a VOC entry that defines the file. DEFINE.DF also creates, in the **&PARTFILES&** file, one record for each part file. The **&PARTFILES&** file is in the UV account.

If a part file belongs to more than one distributed file, the part number of the file must be the same in all distributed files, and all distributed files to which a part file belongs must use the same algorithm.

NLS Mode

The map name for each part file must be the same to avoid data loss. For more information about naming maps, see the *UniVerse NLS Guide*.

Specifying the Partitioning Algorithm

You must specify the partitioning algorithm when you create a distributed file. UniVerse uses the partitioning algorithm to determine in which part file a record belongs.

The partitioning algorithm can be an I-type expression or an external routine, or it can be system-defined. Use the INTERNAL keyword to specify an I-type expression, use the EXTERNAL keyword to specify an external routine, and use the SYSTEM keyword to specify the default partitioning algorithm.

The partitioning algorithm must return a part number. The part number must be a nonzero integer. An I-type expression should use values stored in the @ID system variable to return a part number. The expression cannot access any other data in the record.

Use the following syntax to specify the partitioning algorithm:

Parameter	Description								
INTERNAL	Specifies an I-type expression as the partitioning algorithm. The INTERNAL clause has the following syntax: INTERNAL { [[DATA] <i>filename</i>] <i>record.ID</i> <i>expression</i> } <table><tr><td>DATA</td><td>Specifies that the algorithm expression is stored in the data file of <i>filename</i>. If you do not use the DATA keyword, the expression is stored in the dictionary of <i>filename</i>.</td></tr><tr><td><i>filename</i></td><td>The name of the file containing the algorithm expression.</td></tr><tr><td><i>record.ID</i></td><td>The ID of the record containing the algorithm expression. If you do not specify <i>filename</i>, the record is assumed to be an entry in the VOC file.</td></tr><tr><td><i>expression</i></td><td>The I-type expression used as the partitioning algorithm. Enclose <i>expression</i> in quotation marks if it contains spaces.</td></tr></table>	DATA	Specifies that the algorithm expression is stored in the data file of <i>filename</i> . If you do not use the DATA keyword, the expression is stored in the dictionary of <i>filename</i> .	<i>filename</i>	The name of the file containing the algorithm expression.	<i>record.ID</i>	The ID of the record containing the algorithm expression. If you do not specify <i>filename</i> , the record is assumed to be an entry in the VOC file.	<i>expression</i>	The I-type expression used as the partitioning algorithm. Enclose <i>expression</i> in quotation marks if it contains spaces.
DATA	Specifies that the algorithm expression is stored in the data file of <i>filename</i> . If you do not use the DATA keyword, the expression is stored in the dictionary of <i>filename</i> .								
<i>filename</i>	The name of the file containing the algorithm expression.								
<i>record.ID</i>	The ID of the record containing the algorithm expression. If you do not specify <i>filename</i> , the record is assumed to be an entry in the VOC file.								
<i>expression</i>	The I-type expression used as the partitioning algorithm. Enclose <i>expression</i> in quotation marks if it contains spaces.								

Partitioning Algorithm Syntax

Parameter	Description
	Fields referred to by the I-type expression must be in the same file as the record containing the I-descriptor. If you enter <i>expression</i> on the command line, referenced fields must be in the VOC file. Use the system variable, @ID, to reference record IDs in the distributed file you are defining.
EXTERNAL	Specifies a routine as the partitioning algorithm that is external to UniVerse. The EXTERNAL clause has the following syntax: EXTERNAL <i>routine</i> <i>routine</i> is the external routine used as the partitioning algorithm.
SYSTEM [<i>s</i>]	Specifies the default partitioning algorithm. <i>s</i> is the character used as a separator in record IDs. If you do not specify <i>s</i> , a hyphen (-) separator is the default.

Partitioning Algorithm Syntax (Continued)

Examples

This example defines a distributed file called ORDERS. The ORDERS file comprises two UniVerse files, ORD.USA and ORD.CANADA. The file uses the default partitioning algorithm.

```
>DEFINE.DF ORDERS ORD.USA 1 ORD.CANADA 2 SYSTEM
Creating file "D_ORDERS" as Type 30.
Added "@ID", the default record for Retrieve, to "ORDERS".
Loading default SYSTEM algorithm into DICT ORDERS
Compiling "@PART.ALGORITHM".
FIELD ( @ID , - , 1 )
Part file "ORD.USA", Path "/u2/accts/ORD.USA", Part number 1 Added.
Part file "ORD.CANADA", Path "/u2/accts/ORD.CANADA", Part number 2
Added.
```

The next example adds a third UniVerse file to the ORDERS distributed file:

```
>DEFINE.DF ORDERS ADDING ORD.FRANCE 3
Part file "ORD.FRANCE", Path "/u2/accts/ORD.FRANCE", Part number 3
Added.
```

The next example removes the part file ORD.MEXICO from the ORDERS distributed file, retaining the part number and partitioning algorithm in the ORD.MEXICO file:

```
>DEFINE.DF ORDERS REMOVING ORD.MEXICO RETAIN
Part file "ORD.MEXICO", Path "/u2/accts/ORD.MEXICO", Part number 4
Removed.
```

The next example changes the partitioning algorithm of the ORDERS file. The user-defined algorithm is an I-type expression stored as the record ORD.ALG in the VOCLIB file.

```
>DEFINE.DF ORDERS INTERNAL VOCLIB ORD.ALG
```

DEL.RFILE

Use DEL.RFILE to delete a series of transaction logging log files that have been restored and rolled forward. Deleting log files that have been rolled forward frees up disk space so you can restore more log files. You must be a UniVerse Administrator logged in to the UV account to use DEL.RFILE.

Syntax

DEL.RFILE *first.log last.log* [*logdir.path*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>first.log</i>	The number of the first log file to delete.
<i>last.log</i>	The number of the last log file to delete.
<i>logdir.path</i>	The absolute path of the log directory containing the restored and rolled-forward log files. If you do not specify a path, the log directory defined by the LOGS.DIR record in the dictionary of the UV_LOGS file is used.

DEL.RFILE Parameters

Description

If any of the log files you specify have entries in the UV_LOGS file, DEL.RFILE can delete them only if their status is Released. If the log files do not have entries in UV_LOGS, DEL.RFILE deletes them. DEL.RFILE does not change anything in the UV_LOGS file.

Example

This example deletes four log files from */u2/logrestore*:

```
>DEL.RFILE 349 352 /u2/logrestore
Files   lg349 lg350 lg351 lg352 have been removed
>
```

DELETE

Use DELETE to remove records from a file.

Syntax

DELETE [**DICT**] *filename* [*records*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Deletes records from the file dictionary. If you do not specify DICT, records from the data file are deleted.
<i>filename</i>	The name of the file from which records are deleted.
<i>records</i>	The IDs of records to be deleted, separated by spaces. Do not specify <i>records</i> if select list 0 is active.

DELETE Parameters

Description

You can use select list 0 to specify records to delete. If select list 0 is active, DELETE displays the following messages:

```
You have an active SELECT list.  
Do you wish to DELETE the records previously SELECTed?  
The first record ID = "10002".  
Enter Y or N:
```

DELETE displays only the first record ID. Enter **Y** to delete all the records specified by the select list. Enter **N** to keep the records.

Examples

```
>DELETE PAYABLES 4108 6100
```

```
2 records DELETED.
```

```
>SELECT PAYABLES WITH STATE EQ "MA"
```

```
2 record(s) selected to SELECT list #0.
```

```
>>DELETE PAYABLES
```

```
You have an active SELECT list.
```

```
Do you wish to DELETE the records previously SELECTed?
```

```
The first record ID = "5205".
```

```
Enter Y or N: Y
```

```
2 records DELETED.
```

DELETE.CATALOG

Use DELETE.CATALOG to remove globally, normally, or locally cataloged programs.

Syntax

DELETE.CATALOG *catalog.name*

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>catalog.name</i>	The name of the program to be removed from the catalog space. Catalog names can include only ASCII characters, no multibyte characters. If the program is not in the system catalog, DELETE.CATALOG searches the VOC file for a local catalog entry. If the program is locally cataloged, only the VOC entry is deleted, not the object code.

DELETE.CATALOG Parameter



Note: *If a job is using the cataloged program when you delete it, the deletion does not affect the job unless it tries to reload the program.*

Examples

This example deletes a normally cataloged program from the catalog space:

```
>DELETE.CATALOG *SALES*UPDATE
"*SALES*UPDATE" DELETED from the system CATALOG.
>
```

The next example deletes a globally cataloged program from the catalog space:

```
>DELETE.CATALOG *UPDATE
"*UPDATE" DELETED from the system CATALOG.
>
```

The next example deletes a locally cataloged program from the VOC file:

```
>DELETE.CATALOG TEST
"*SALES*TEST" is not in the CATALOG space.
LOCAL catalog entry "TEST" DELETED from your VOC.
>
```

DELETE.ENCRYPTION.KEY

Use the DELETE.ENCRYPTION.KEY command to delete a key from a key store.

Syntax

DELETE.ENCRYPTION.KEY [FORCE] *key.id* [*password*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
FORCE	Forces the encryption key to be deleted, even if it is referenced by an encrypted record or field.
<i>key.id</i>	The encryption key to delete.
<i>password</i>	The password for the encryption key to delete.

DELETE.ENCRYPTION.KEY Parameters

Description

You must be the owner of the file or logged on as root or a UniVerse Administrator to delete an encryption key, and you must provide the correct password. If the key is referenced by any encrypted field or file, deleting the key will fail, unless you specify FORCE.

DELETE.ENCRYPTION.WALLET

Use the DELETE.ENCRYPTION.WALLET command to delete an encryption wallet from the key store.

Syntax

DELETE.ENCRYPTION.WALLET [FORCE] <wallet_id> <wallet_password>

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
FORCE	Forces the encryption wallet to be deleted, even if it is referenced by an encrypted record or field
wallet_ID	The ID of the encryption wallet you want to delete.
wallet_password	The password for the encryption wallet you want to delete.

DELETE.ENCRYPTION.KEY Parameters

DELETE.FILE

Use DELETE.FILE to delete the VOC file entry for a UniVerse file. You can delete the data file, the file dictionary, or both.

Syntax

DELETE.FILE [DICT | DATA] [*filename*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Deletes only the file dictionary.
DATA	Deletes only the data file.
<i>filename</i>	The name of the file to delete. If you do not specify <i>filename</i> , DELETE.FILE uses select list 0 if it is active. The record IDs in the select list must be UniVerse file names.

DELETE.FILE Parameters

Description

If you do not specify the DATA or DICT keyword, DELETE.FILE deletes the data file and its dictionary.

If the data file path in the VOC file entry is the same as *filename*, or if the dictionary path is D_*filename*, DELETE.FILE deletes the file. If the data or dictionary paths are anything else, DELETE.FILE displays a warning message and asks if you really want to delete the files listed in the VOC entry.

If you specify DELETE.FILE while a select list is active, DELETE.FILE deletes all files specified in the select list.

Be careful when you delete a file that has synonyms or remote file pointers because DELETE.FILE has no effect on those synonyms or remote file pointers. Use the DELETE command to delete file synonyms and remote file pointers from the VOC file after you have deleted the file.

If *filename* is a remote file, DELETE.FILE deletes only the VOC file record that points to it. DELETE.FILE displays the path of the remote file.

Do not use DELETE.FILE to delete a distributed file. Instead, use the [DEFINE.DF](#) command to remove all part files. When DEFINE.DF removes the last part file, it also deletes the distributed file.

To delete a part file of a distributed file, first use DEFINE.DF to remove the part file from the distributed file, then use DELETE.FILE to delete the file. If a former part file still retains its part number and partitioning algorithm, you must cancel them with DEFINE.DF, then use DELETE.FILE.



***Note:** You cannot use DELETE.FILE to delete an SQL table. Use the SQL statement DROP TABLE.*

Examples

This example deletes the data file and the file dictionary of the INVENTORY.92 file:

```
>DELETE.FILE INVENTORY.92
DELETED file "NVENTORY.92", Type 18, Modulo 23.
DELETED file "D_INVENTORY.92", Type 3, Modulo 1.
DELETED file definition record "INVENTORY.92" in the VOC file.
```

The next example deletes only the data file of INVENTORY.92:

```
>DELETE.FILE DATA INVENTORY.92
DELETED file "INVENTORY.92", Type 18, Modulo 23.
Field 2 in file definition record "INVENTORY.92" has been set to
nothing.
```

In the next example, INV is an F-type synonym for INVENTORY. DELETE.FILE asks if the user wants to delete the data file or the file dictionary, and the user answers **N** to both. DELETE.FILE then deletes the VOC entry for INV.

```
>DELETE.FILE INV
DATA entry "INVENTORY" does not match expected DATA "INV".
Do you wish to DELETE this DATA file (Y/N)? N
DICT entry "D_INVENTORY" does not match expected DICT "D_INV".
Do you wish to DELETE this DICT file (Y/N)? N
DELETED file definition record "INV" in the VOC file.
```

In the next example, the file definition (F-type) for SALES.EAST defines a file in another account. DELETE.FILE does not delete the data file or the file dictionary, but it does delete the VOC entry that points to the remote files.

```
>DELETE.FILE SALES.EAST  
Cannot DELETE a remote file.  
Cannot DELETE a remote file.  
DELETED file definition record "SALES.EAST" in the VOC file.
```

DELETE.INDEX

Use DELETE.INDEX to delete a secondary index from a file.

Syntax

```
DELETE.INDEX [ DICT ] [ filename ] [ indexes | ALL ]
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
<i>filename</i>	The name of a UniVerse file from which to delete a secondary index. If you do not specify <i>filename</i> , DELETE.INDEX prompts for it.
<i>indexes</i>	A list of index names separated by spaces. <i>indexes</i> are the names of fields in <i>filename</i> used as secondary index keys.
ALL	Specifies to delete all the secondary index keys in the file.

DELETE.INDEX Parameters

Description

If you do not specify either *indexes* or ALL, DELETE.INDEX prompts you to enter an index name.

DELETE.LFILE

Use DELETE.LFILE to delete empty log files from the transaction logging log directory. You must be a UniVerse Administrator logged in to the UV account to use DELETE.LFILE.

Syntax

DELETE.LFILE *files*

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>files</i>	The number of log files you want to delete from the log directory. DELETE.LFILE deletes only files whose status is Available.

DELETE.LFILE Parameter

Description

DELETE.LFILE verifies that each log file is empty, then deletes the log file and the corresponding record in the UV_LOGS file, and reduces the value of the LOG.NEXT record in the dictionary of UV_LOGS.

Example

This example deletes four empty log files from the log directory:

```
>DELETE.LFILE 4
```

```
Deleting lg356
```

```
Deleting lg355
```

```
Deleting lg354
```

```
Deleting lg353
```

```
>
```

DELETE.LIST

Use DELETE.LIST to remove a saved select list that was created with the SAVE.LIST command, or to remove a sentence stack that was saved by logging out.

Syntax

DELETE.LIST [*listname* | *]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>listname</i>	The record ID of the saved list in the &SAVEDLISTS& file. If you do not specify <i>listname</i> , DELETE.LIST deletes a list called &TEMPport#& . See the SAVE.LIST command for information about this <i>listname</i> .
*	Specifies all <i>listnames</i> in the &SAVEDLISTS& file. DELETE.LIST clears the &SAVEDLISTS& file, leaving only your saved sentence stacks.

DELETE.LIST Parameters

Examples

To delete the list called DECEMBER, enter the following:

```
>DELETE.LIST DECEMBER
Saved list "DECEMBER" DELETED.
```

To delete a list saved without a name, enter the following:

```
>DELETE.LIST
Saved list "&TEMP-11057&" DELETED.
```

DISABLE.DECRYPTON

Use the DISABLE.DECRYPTON command to turn off decryption on a field you specify.

Syntax

DISABLE.DECRYPTON *filename* <*field_list*>

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>filename</i>	The name of the file on which you want to disable decryption.
<i>field_list</i>	A comma-separated list of fields for which you want to disable decryption. Do not enter spaces between the field names.

DISABLE.DECRYPTON Parameters

DISABLE.INDEX

Use DISABLE.INDEX to disable automatic updating of the secondary key indexes of a file.

Syntax

DISABLE.INDEX [DICT] [*filenames*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
<i>filenames</i>	The names of the UniVerse files for which to disable secondary index updating. If you do not specify <i>filenames</i> , DISABLE.INDEX prompts for them.

DISABLE.INDEX Parameters

Description

After you build an index, automatic updating is enabled. Disabling automatic updating can increase processing speed when entering data. However, reports generated from files while automatic updating is disabled can contain incorrect information. That is, records added after updating is disabled cannot be accessed using a secondary key, deleted records can still be accessed using a secondary key, and modified records can be associated with the wrong secondary key.

Use [ENABLE.INDEX](#) to reenable automatic indexing. The ENABLE.INDEX command does not update index entries. To make the index up-to-date, issue the [UPDATE.INDEX](#) command. It is good practice to issue an UPDATE.INDEX command after you use the ENABLE.INDEX command.

Users who have the specified file open when DISABLE.INDEX is issued continue to have indexes updated.

Use [LIST.INDEX](#) to display the updating status of an index.

DISPLAY

Use DISPLAY to display a line of text on the terminal.

Syntax

DISPLAY *text*

Parameter

The following table describes each parameter of the syntax.

Parameter	Description
<i>text</i>	The text that appears on the terminal.

DISPLAY Parameter

Example

This example uses DISPLAY in a paragraph:

```
BALANCES
0001: PA
0002: DISPLAY PRINTING VENDOR LIST
0003: LIST PAYABLES WITH BAL.DUE = 0 LPTR
0004: DISPLAY PRINTING BALANCES
0005: LIST PAYABLES WITH BAL.DUE >= 10 LPTR
```

When you run this paragraph, it displays the following messages on the screen:

```
PRINTING VENDOR LIST
PRINTING BALANCES
```

DIVERT.OUT

Use DIVERT.OUT to begin diverting the output of subsequent commands to a record in a type 1 or type 19 file. Output continues to appear on the terminal unless you turn it off.

Syntax

DIVERT.OUT *action* [*filename record*] [*parameter*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>filename</i>	The name of an existing type 1 or type 19 file.
<i>record</i>	The name of a record in <i>filename</i> where the output is to be diverted.

DIVERT.OUT

action can be one of the following:

Action	Description
ON	Starts sending output to specified <i>record</i> in <i>filename</i> .
OFF	Stops copying terminal output.
FILE.OFF	Suspends current output diversion. Subsequent command output is not diverted until you enter a DIVERT.OUT FILE.ON command.
FILE.ON	Resumes output diversion previously suspended by the DIVERT.OUT FILE.OFF command.
TTY.OFF	Turns off terminal (tty) display of output and user input.
TTY.ON	Turns on terminal (tty) output display if it is disabled.

DIVERT.OUT Actions

parameter can be any of the following keywords:

<i>parameter</i>	Description
TTY.OFF	Turns off terminal (tty) display of output and user input.
TTY.ON	Turns on terminal (tty) output display.
TRUNCATE	Specifies that an existing record be replaced by the current diversion. Truncates any part of the existing record not overwritten by the current diversion.
APPEND	Adds current output diversion to the end of an existing record.

DIVERT.OUT *parameters*

Description

DIVERT.OUT works like the [COMO](#) command, except that COMO automatically creates a [&COMO&](#) file if it does not exist, and COMO time-stamps the beginning and end of a COMO session.

If *record* already exists, specify either the [APPEND](#) keyword to append new output to the end of the record or the [TRUNCATE](#) keyword to completely overwrite the existing record.

COMO and DIVERT.OUT commands cannot be nested within one another.

Examples

```
>CREATE.FILE DIVERSION 1
Creating file "DIVERSION" as Type 1.
Creating file "D_DIVERSION" as Type 3, Modulo 1, Separation 2.
Added "@ID", the default record for Retrieve, to "D_DIVERSION".
>DIVERT.OUT ON DIVERSION FIRST
>LIST SUN.MEMBER WITH FNAME EQ "EDGAR"
LIST SUN.MEMBER WITH FNAME EQ "EDGAR" 11:08:15am 20 Oct 1995
PAGE 1
MEMBER ID. FIRST NAME LAST NAME. YEAR JOINED
INTERESTS.....

4309 EDGAR WILLIAMS 1984 FISHING
SAILING

1 records listed.
>DIVERT.OUT FILE.OFF
```

```
>TIME
11:08:16 20 OCT 1995
>DIVERT.OUT FILE.ON
>DATE
Wednesday, October 20, 1995 11:08am
>DIVERT.OUT OFF
```

This example establishes the record FIRST in the type 1 file named DIVERSION. Its contents are as follows:

```
001: >LIST SUN.MEMBER WITH FNAME EQ "EDGAR"
002:
003: LIST SUN.MEMBER WITH FNAME EQ "EDGAR" 11:08:16am 20 Oct 1995
PAGE 1
004: MEMBER ID.      FIRST NAME      LAST NAME.      YEAR JOINED
INTERESTS..
005:
006: 4309            EDGAR            WILLIAMS            1984
FISHING
007:
SAILING
008:
009: 1 records listed.
010: >DIVERT.OUT FILE.OFF
011: >DATE
012: Wednesday, October 20, 1995 11:08am
013: >DIVERT.OUT OFF
```

DLIST

Use DLIST to display a listing of I-descriptor object code. DLIST displays the I-type expression followed by the lines of compiled code it generated. DLIST also displays statistics about the I-descriptor.

Syntax

DLIST *filename I-descriptor*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>filename</i>	The name of the file whose dictionary contains <i>I-descriptor</i> .
<i>I-descriptor</i>	The record ID of the I-descriptor in the dictionary of <i>filename</i> .

DLIST Parameters

Example

```
>DLIST TESTFILE WO.CODE.NUM

Item "WO.CODE.NUM": CONVERT("ABC", "234", WO.CODE); IF NUM(@1)
THEN @1 ELSE 0; IF @ 2 THEN @2 ELSE 0

Compiled 16:51 25 JAN 95
3 expressions
0 TOTALS
5 constants

0000: 060 dyn_extract      @RECORD 3 0 0  => $R0
000C: 03C convert          "ABC" "234" $R0  => @1
0016: 110 num              @1  => $R0
001C: 1A0 testf            $R0 0030:
0024: 0F8 move             @1  => @2
002A: 0C2 jump             0036:
0030: 0F8 move             0   => @2
```

0036: 1A0 testf	@2 004C:
0040: 0F8 move	@2 => @3
0046: 0C2 jump	0052:
004C: 0F8 move	0 => @3
0052: 0F8 move	@3 => @ANS
0058: 15C return	

DOS

Use DOS to invoke a DOS command shell from within UniVerse.

Syntax

DOS [*/c command* | *pathname*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>command</i>	Specifies the command to execute.
<i>pathname</i>	The path of a batch file or command file to execute.

DOS Parameters

Description

UNIX. The DOS command is supported only if you use a suitable emulator.

Example

```
>DOS /c DIR
```

ED

Use ED to invoke the UniVerse Editor.

Syntax

ED [**DICT**] [*filename*] [*records* | *]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Edits records in the file dictionary.
<i>filename</i>	The name of an existing file.
<i>records</i>	The record IDs of the records you want to edit. Separate multiple records by spaces. You can also use an active select list. Do not specify an empty string as the record ID.
*	Specifies all records in the file.

ED Parameters

Description

The Editor creates records, not files. You must first create files with [CREATE.FILE](#).

Use the UniVerse Editor to create BASIC programs and create, change, or delete records in data files and file dictionaries. Each line listed by the Editor shows one field in a record.

If you enter ED with no qualifiers, UniVerse prompts for *filename* and a record ID. If you enter ED *filename*, UniVerse prompts for a record ID. If you do not specify *filename* and select list 0 is active, ED uses the select list.

EDIT.CONFIG

Use EDIT.CONFIG in NLS mode to display and edit UniVerse configurable parameters. You must be a UniVerse Administrator logged in to the UV account to use this command.

Syntax

EDIT.CONFIG

Description

When you enter EDIT.CONFIG, you see a screen similar to this:

```
1  >EDIT.CONFIG

EDIT.CONFIG - UniVerse Configuration Tool                      13
February 1997
1
(using: ./uvconfig)
1  Parameter                                Value    Description
1  PSEMNUM                                *400
2  FLTABSZ                                11
3  GLTABSZ                                300
4  RLTABSZ                                300
5  PAKTIME                                300
6  QBREAK                                 1
7  UVSYNC                                 1
8  BLKMAX                                 8192
9  NLSDEFFILEMAP                          CNS1136
10 NLSDEFDIRMAP                          ASCII+MARKS
11 NLSNEWFILEMAP                          UNICODE
12 NLSNEWDIRMAP                          *ISO8859-2    <<NLSNEWDIRMAP - Map to be
assumed
13 NLSDEFGCIMAP                          ASCII        for all new Type 1/19
files
14 NLSDEFSEQMAP                          ISO8859-1+MARKS    created. Default is
ISO8859-1.
15 NLSREADELSE                            0
16 NLSWRITEELSE                            0
17 NLSDEFDEVMAP                          ISO8859-1
18 NLSOSMAP                              ASCII
1  .nn-select <cr>-next P-prev B-pg Fwd R-restore
?-help Q-quit
1  [Original value: ISO8859-1 ] New value:
```

At the top of the screen the display area shows the first page of parameters, their values, and a description of the current parameter. At the bottom of the screen, the input area shows you the commands you can use and prompts you to enter a command or value.

The following list shows all the commands you can use in the configuration editor:

Command	Description
<i>.nn</i>	Selects parameter <i>nn</i> .
<cr>	Moves forward one parameter. At the bottom of the page, scrolls forward half a page.
P	Moves backward one parameter. At the top of the page, scrolls backward half a page.
B	Scrolls backward half a page.
F	Scrolls forward half a page.
R	Restores the value of the current parameter to its previous value when the file was first loaded. Note that a parameter whose value has changed is preceded by an asterisk (*). There are two examples in the example screen.
?	Displays help about the available commands.
!	Refreshes the screen.
/ <i>pattern</i> /	Searches forward through the parameters for <i>pattern</i> , if any.
\ <i>pattern</i> \	Searches backward for <i>pattern</i> if any.
//	Repeats the last search forward (made with / <i>pattern</i> /). The current stored <i>pattern</i> can be found on the help screen.
\\	Repeats the last search (made with \ <i>pattern</i> \).
?M	Displays a list of map IDs from the NLS.MAP.DESCS file and prompts you to choose one. The ID becomes the new value of the current parameter unless you quit from the prompt. Map IDs preceded by an asterisk (*) are in shared memory; you need not build and load them.
Q or X	Quits the configuration editor. If you changed the file, you are prompted to save it.

Configuration Editor Commands



Any other value that you enter at the prompt becomes the new one for the current parameter.

Note: *The prompt for input displays the original value; that is, the value associated with the parameter name when the `uvconfig` file was first read.*

To view the current settings of the parameters, use the [CONFIG DATA](#) command.

EDIT.LIST

Use EDIT.LIST to edit the contents of a select list saved in the file [&SAVEDLISTS&](#).

Syntax

EDIT.LIST [*listname*]

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>listname</i>	The name of the list you want to edit. Separate multiple list names with spaces, or use an active select list.

EDIT.LIST Parameter

Description

EDIT.LIST lets you create, change or add to a list using the UniVerse Editor. EDIT.LIST is the same as [ED](#) using the filename [&SAVEDLISTS&](#) and the record ID of the list you want to edit.

If you do not specify *listname*, UniVerse prompts for it.

If you do not specify *listname* and select list 0 is active, EDIT.LIST uses the contents of the select list as the record IDs of the lists you want to edit.

ENABLE.DECRYPTON

Use the ENABLE.DECRYPTON command to activate encryption on specific fields in a file.

Syntax

ENABLE.DECRYPTON *filename* <*field_list*>

Parameters

The following table describes each parameter of the syntax..

Parameter	Description
<i>filename</i>	The name of the file on which you want to enable decryption.
<i>field_list</i>	A comma-separated list of fields for which you want to enable decryption. Do not enter spaces between the field names.

ENABLE.DECRYPTON Parameters

ENABLE.INDEX

Use ENABLE.INDEX to reenable automatic updating of the secondary key indexes of a file after the updating has been disabled by the [DISABLE.INDEX](#) command.

Syntax

ENABLE.INDEX [DICT] [*filenames*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
<i>filenames</i>	The names of UniVerse files for which to enable secondary index updating. If you do not specify <i>filenames</i> , ENABLE.INDEX prompts for them.

ENABLE.INDEX Parameters

Description

Automatic updating is initially enabled. After disabling it, you can reenable automatic updating with ENABLE.INDEX.

ENABLE.INDEX does not actually update the index entries. Use the [UPDATE.INDEX](#) command for this. You can use UPDATE.INDEX when automatic updating is enabled or when it is disabled.

If an indexed file is open when ENABLE.INDEX is issued, the indexes are not immediately updated. Some inconsistencies could continue even after updating is enabled and the indexes have been updated. See the UPDATE.INDEX command for a method of ensuring that the indexes are up-to-date.

Use [LIST.INDEX](#) to display the updating status of an index.

ENABLE.RECOVERY

Use ENABLE.RECOVERY to enable the transaction logging system. You must be a UniVerse Administrator logged in to the UV account to use this command.

Syntax

ENABLE.RECOVERY [YES | NO] [INFORM]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
YES	Retains the current logging info file.
NO	Removes the current logging info file.
INFORM	Displays messages during the startup process.

ENABLE.RECOVERY Parameters

Description

ENABLE.RECOVERY starts the log daemon *uvlogd*.

You should fully back up your UniVerse files before enabling transaction logging for the first time with ENABLE.RECOVERY or after [SHUTDOWN.RECOVERY](#).

Example

This example starts the log daemon and retains the current logging info file *uvlogd.info*:

```
>ENABLE.RECOVERY YES
Request to Enable Logging Subsystem made at 12:51:52 on 01 OCT
1996. You can use the 'Display logging state' menu to verify the
current state of the logging subsystem.
```



ENCRYPT.FILE

Use the ENCRYPT.FILE command to create a file in which each record is encrypted.

Note: You cannot encrypt an index file.

Syntax

```
ENCRYPT.FILE {<filename> <type> <modulo> <separation> | <30 | dynamic>  
parameter [value]...} <USING partition> < {WHOLERECORD |  
fieldname},alg,key[,pass] [fieldname,alg,key[,pass]]...>
```

Parameters

Most of the ENCRYPT.FILE parameters are the same as the RESIZE command parameters. If the file you are encrypting is empty, you do not need to specify any of the RESIZE parameters. If the file you are encrypting is not empty, and you know that the file needs resizing because encrypting the file will increase the record size, you should specify the RESIZE parameters.

The following table describes each parameter of the syntax.

Parameter	Description
<i>filename</i>	The UniVerse file name. If you do not specify <i>filename</i> , ENCRYPT.FILE prompts for the name. <i>filename</i> must follow the UniVerse naming conventions. For more information about naming conventions, see “File Naming Conventions” in <i>UniVerse User Reference</i> .
<i>type</i>	The UniVerse file type for the file you are encrypting. Type 1 or type 19 files are not hashed and are usually used to store text files such as BASIC programs. Types 2 through 18 are hashed files. Type 25 is a balanced tree file.
<i>modulo</i>	The modulo for the file you are encrypting. The modulo should be an integer from 1 through 8,388,608 defining the number of groups in the file. UniVerse ignores <i>modulo</i> if you specify a nonhashed or dynamic file type.

ENCRYPT.FILE Parameters

Parameter	Description
<i>separation</i>	The separation for the file you are encrypting. The separation should be an integer from 1 through 8,388,608, specifying the group buffer size is 512-byte blocks. UniVerse ignores <i>separation</i> if you specify a nonhashed or dynamic file type.
30	Encrypts a dynamic file.
dynamic	Encrypts a dynamic file.
USING <i>partition</i>	Specifies the path of the work area that ENCRYPT.FILE will use for creating the necessary temporary files. For example, the following command encrypts SUN.MEMBER as a dynamic file, and creates the temporary files it needs in the partition <i>/u4</i> : >ENCRYPT.FILE SUN.MEMBER DYNAMIC USING /u4 ENCRYPT.FILE moves the files back into the correct directory after encrypting the SUN.MEMBER file.
WHOLERECORD	Specifies to fully encrypt every record in the file.
<i>fieldname,alg,key,pass</i>	Specifies the field name to encrypt, and the algorithm, key, and password to use. You can use a different algorithm and key for each field. If you do not specify a password, but created the key using password protection, UniVerse prompts for the password. If several fields use the same password, you only have to specify it once, at the first field that uses that key. <i>fieldname</i> The name of the field to encrypt. <i>alg</i> The algorithm to use for encryption. See “UniVerse Encryption Algorithms” on page 9 for a list of valid values. <i>key</i> The key ID to use for the field encryption. <i>pass</i> The password corresponding to the <i>key</i>.

ENCRYPT.FILE Parameters (Continued)

Specify the following parameters only for dynamic files:

Parameter	Description
GENERAL	Specifies the general hashing algorithm for a dynamic file. GENERAL is the default.
SEQ.NUM	Specifies a hashing algorithm suitable for sequential numbers for a dynamic file. Use this hashing algorithm only for records with IDs that are mainly numeric, sequential, and consecutive.
GROUP.SIZE { 1 2 }	Specifies the size of each group in the file, either 1 or 2. 1 specifies a group size of 2048 bytes, which is equivalent to a separation of 4. 2 specifies a group size of 4096 bytes, which is equivalent to a separation of 8. A group size of 2048 (GROUP.SIZE 1) is the default.
MINIMUM.MODULUS <i>n</i>	Specifies the minimum modulo of the file, an integer value greater than 1. This value is also the initial value of the modulo of the dynamic file. A minimum modulo of 1 is the default.
SPLIT.LOAD <i>n</i>	Specifies the level at which the file's modulo is increased by 1. SPLIT.LOAD takes a numeric argument indicating the percentage of space allocated for the file. When the data in the file exceeds the specified percentage of the space allocated for the file, the data in one of the groups is divided equally between itself and a new group, to increase the modulo by 1. The default SPLIT.LOAD is 80%.
MERGE.LOAD <i>n</i>	Specifies the level at which the file's modulo is decreased by 1. MERGE.LOAD takes a numeric argument indicating the percentage of space allocated for the file. When the data in the file is less than the specified percentage of the space allocated for the file, the data in the last group of the file is merged with another group, to decrease the modulo by 1. The default MERGE.LOAD is 50%.

ENCRYPT.FILE Parameters for Dynamic Files

Parameter	Description
LARGE.RECORD <i>n</i>	Specifies the size of a record considered too large to be included in the primary group buffer, specified as an integer or a percentage. Specified as an integer, the value is the number of bytes a record must contain to be considered a large record. Specified as a percentage, the value is a percentage of the group size. When the size of a record exceeds the specified value, the data for the record is put in an overflow buffer, but the record ID is put in the primary buffer. This method of large record storage increases access speed. The default LARGE.RECORD size is 80%.
RECORD.SIZE <i>n</i>	Calculates the values for group size and large record size based on the value of the estimated average record size specified. The value is your estimate of the average record size for the dynamic file, specified in bytes. RECORD.SIZE does not limit the size of records. If you specify a value for group size (GROUP.SIZE) or for large record size (LARGE.RECORD), those values override the value calculated by RECORD.SIZE.
MINIMIZE.SPACE	Calculates the best amount of space required by the file (at the expense of access time), using the values for the split load, merge load, and large record size. If you specify values for split load, merge load, or large record size, those values override the value calculated by MINIMIZE.SPACE. If you specify MINIMIZE.SPACE and RECORD.SIZE, the value for large record size calculated by MINIMIZE.SPACE is used above the value calculated by RECORD.SIZE.

ENCRYPT.FILE Parameters for Dynamic Files (Continued)

Description

Encrypting a file requires exclusive access to the file, and is very time consuming. During the encryption process, UniVerse creates a temporary file and writes the newly encrypted data to that file. If any errors occur during the encryption process, the command aborts and the original file is left intact.

ENVIRONMENT or ENV

Use ENVIRONMENT or ENV to set and display environment variables.

Syntax

ENV[**IRONMENT**] [**DISPLAY**]

ENV[**IRONMENT**] [**SET**] *env.variable=value*

ENV[**IRONMENT**] **CLEAR** *env.variable*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DISPLAY	Lists the current environment settings.
SET	Sets the specified environment variable to <i>value</i> .
<i>env.variable</i>	The name of the environment variable you want to set or clear. <i>env.variable</i> names are case-sensitive.
<i>value</i>	The value to which you are setting <i>env.variable</i> . <i>value</i> can be of zero length.
CLEAR	Unsets the specified environment variable. If <i>env.variable</i> is not set, the CLEAR keyword does nothing.

ENVIRONMENT Parameters

Description

ENV prints variables containing control characters (that is, ASCII characters 0 through 26) as a caret (^) followed by the character. For example:

```
BELL=^G
```

Setting a variable to a zero-length value is different from unsetting it. For detailed information about environment variables, see your operating system documentation.

Examples

The ENVIRONMENT command with no options prints the current environment settings:

```
>ENVIRONMENT
LEVEL=-
TZ=EST5EDT
USER=todd
PATH=./usr/ardent/uv/bin:/bin:/src/upix/sp:/usr/bin:/usr/sbin:/etc
SHELL=/bin/csh
HOME=/usr/todd
```

Each of the next two examples sets the environment variable DISPLAY to “romero”:

```
>ENV DISPLAY=romero
>ENVIRONMENT SET DISPLAY=romero
```

The next example sets the DISPLAY variable to an empty string:

```
>ENV DISPLAY=
```

The next example unsets the DISPLAY variable:

```
>ENV CLEAR DISPLAY
```

ESEARCH

Use ESEARCH to create a select list of records that contain an occurrence of a specified string. ESEARCH is a synonym for the [SEARCH](#) command.

Syntax

```
ESEARCH [ DICT | USING [ DICT ] dictname ] filename [ records | FROM n ]  
[ selection ] [ output.limiter ] [ sort ] [ TO n ] [ options ]
```

EXCHANGE

Use EXCHANGE to switch the contents of one record with another record in a file.

Syntax

EXCHANGE [**DICT**] *filename record.a record.b*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
<i>filename</i>	The name of the file containing the records to be switched.
<i>record</i>	The name of a record whose contents will be switched with the contents of the other record.

EXCHANGE Parameters

Description

The EXCHANGE command is a proc that copies the contents of *record.a* into a temporary file, then copies the contents of *record.b* into *record.a*, and then copies the contents of the temporary file into *record.b*.

If one of the records does not exist in *filename*, EXCHANGE changes the name of the existing record to the name of the other record.

FANCY.FORMAT

Use FANCY.FORMAT to format BASIC source statements into a logical block structure by indenting lines and inserting space around arithmetic operators so that the program is easier to read.

Syntax

FANCY.FORMAT [*filename*] [*program* | *] [-LIST]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>filename</i>	The name of the file containing the BASIC programs.
<i>program</i>	The name of the BASIC program. If select list 0 is active, you need not specify <i>program</i> .
*	Specifies all programs in the file.
-LIST	Prints the program on the printer.

FANCY.FORMAT Parameters

Description

If you do not specify *filename* or *program*, FANCY.FORMAT prompts for them.

FANCY.FORMAT is like [FORMAT](#), but it handles spacing on each line differently. For example, FANCY.FORMAT adds a space around the = sign.

FILE.STAT

Use FILE.STAT to get statistical information about static hashed files. FILE.STAT displays the file type, modulo, number of bytes in the file, number of records in the file, and number of groups in the file. The command also displays averages and lists the minimum and maximum number of bytes in a record.

Syntax

FILE.STAT [DICT] *filename* [[DICT] *filename*] ... [LPTR]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
<i>filename</i>	The name of a UniVerse hashed file.
LPTR	Sends output to the printer.
FILE.STAT Parameters	

Description

If you specify a type 1, type 19, type 25, or dynamic file (type 30), an error message appears.

If FILE.STAT does not provide the information that you need, try [GROUP.STAT](#), [GROUP.STAT.DETAIL](#), [HASH.AID](#), [HASH.HELP](#), [HASH.HELP.DETAIL](#), [HASH.TEST](#), or [HASH.TEST.DETAIL](#). The GROUP.STAT commands report more details about data distribution in the file. The HASH.HELP commands recommend the file type, modulo, and separation, and the optimal combination for the file. The HAST.TEST commands let you test new file sizes. HASH.AID lets you test the file size and show additional file size information.

Use [ANALYZE.FILE](#) to get statistical information about dynamic files.

NLS Mode

The report includes the name of the file's map.

Example

The following listing is the output of FILE.STAT for the SUN.MEMBER data file:

```
>FILE.STAT SUN.MEMBER
File name                      = SUN.MEMBER
File type                      = 2
NLS Character Set Mapping      = ISO8859-
1+MARKS(NLSDEFFILEMAP)
Number of groups in file (modulo) = 2
Separation                     = 2
Number of records              = 13
Number of physical bytes       = 3072
Number of data bytes           = 1240

Average number of records per group = 6.5000
Average number of bytes per group   = 620.0000
Minimum number of records in a group = 6
Maximum number of records in a group = 7

Average number of bytes per record  = 95.3846
Minimum number of bytes in a record = 80
Maximum number of bytes in a record = 120

Average number of fields per record = 10.0000
Minimum number of fields per record = 10
Maximum number of fields per record = 10

Groups   25%    50%    75%   100%   125%   150%   175%   200% full
         0      0      2      0      0      0      0      0
```

FILE.USAGE

Use FILE.USAGE to list file usage statistics for any static hashed file.

Syntax

FILE.USAGE [DICT] [*filenames*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
<i>filenames</i>	A list of filenames separated by spaces. If you do not specify <i>filenames</i> , FILE.USAGE prompts for them. If a select list is active, you need not specify <i>filename</i> .

FILE.USAGE Parameters

Description

The FILE.USAGE report contains the following data:

Column Heading	Description
Reads for update	The number of locked reads attempted of the file. The BASIC READU statement increments this field. This field also displays the percentage of total reads.
Blocked reads	The number of locked reads of the file that fail because the file is locked by another user. This field also displays the percentage of total reads.
Read sharelocks	The number of locked reads attempted of the file. The BASIC READL statement increments this field. This field also displays the percentage of total reads.

FILE.USAGE Data

Column Heading	Description
Blocked sharelocks	The number of locked reads of the file that fail because the file is exclusively locked by another user. This field also displays the percentage of total reads.
Oversized reads	The number of reads of the file of data records too large to fit in a single group buffer. This field also displays the percentage of total reads.
Total reads	The sum of all reads of the file.
Writes for update	The number of writes to the file that preserve the update lock on the file. The BASIC WRITEU statement increments this field. This field also displays the percentage of total writes.
Update writes	The number of writes to the file while the file had an update lock set. This field also displays the percentage of total writes.
Blocked writes	The number of writes to the file that fail because the file is locked by another user. This field also displays the percentage of total writes.
Oversized writes	The number of writes to the file of data too large to fit in a single group buffer. This field also displays the percentage of total writes.
Total writes	The sum of all writes to the file.
Total selects	The number of times a select has been performed on the file. If the select was performed using a secondary index, this field is incremented for the index file, but not for the data file.
Total deletes	The number of times a delete has been performed on the file.
Total clears	The number of times the file has been cleared.
Total opens	The number of times the file has been opened.
Overflow buffer scans	The number of times overflow group buffers have been accessed in the file.
Buffer compactions	The number of space recovery group buffer compactions done in the file.

FILE.USAGE Data (Continued)

A file's usage statistics are reset by a RESIZE operation or by [FILE.USAGE.CLEAR](#). Use the FILE.USAGE.CLEAR command to start collecting file usage statistics on a file. Use [FILE.USAGE.OFF](#) to disable collection of file usage statistics. You must have write permission on a file to collect its statistics.

FILE.USAGE.CLEAR

Use FILE.USAGE.CLEAR to reset the file usage statistics displayed by the [FILE.USAGE](#) command. You must have write permission on a file to collect its statistics.

Syntax

FILE.USAGE.CLEAR [DICT] [*filenames*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
<i>filenames</i>	A list of file names separated by spaces. If you do not specify <i>filenames</i> , FILE.USAGE.CLEAR prompts for them. If a select list is active, you need not specify <i>filenames</i> .

FILE.USAGE.CLEAR Parameters

FILE.USAGE.OFF

Use FILE.USAGE.OFF to disable collection of file usage statistics displayed by the [FILE.USAGE](#) command. You must have write permission on a file to use this command.

Syntax

FILE.USAGE.OFF [DICT] [*filenames*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
<i>filenames</i>	A list of file names separated by spaces for processing. If you do not specify <i>filenames</i> , FILE.USAGE.OFF prompts for them. If a select list is active, you need not specify <i>filenames</i> .

FILE.USAGE.OFF Parameters

FORM.LIST

Use FORM.LIST to create a select list from a list of data elements stored in a record. A select list created by FORM.LIST is the same as one created by [SELECT](#) or [SSELECT](#).

Syntax

FORM.LIST [*filename*] [*record*] [TO *n*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>filename</i>	The file containing <i>record</i> .
<i>record</i>	The record ID of the record whose elements are to make up the select list.
TO <i>n</i>	Creates a numbered select list. <i>n</i> specifies the list number from 0 through 10. If you omit the TO clause, FORM.LIST creates select list 0.

FORM.LIST Parameters

Description

If you enter FORM.LIST without specifying *filename* or *record*, FORM.LIST prompts for them.

You can use a BASIC program, an editor, or [REVISE](#) to create the record containing list elements. Separate elements with newlines if the file is a type 1 or type 19 file. Separate elements with field marks if the file is hashed.

Example

This example creates an active select list from the record LIST1 in the file LISTS, and waits for a command to use the select list:

```
>FORM.LIST LISTS LIST1
```

```
126 record(s) selected to SELECT list #0
```

```
>>
```

FORMAT

Use FORMAT to format BASIC source statements into a logical block structure by indenting lines so that the program is easier to read.

Syntax

FORMAT [*filename*] [*program* | *] [-LIST]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>filename</i>	The name of the file containing the BASIC programs.
<i>program</i>	The name of the BASIC program. If select list 0 is active, you need not specify <i>program</i> .
*	Specifies all programs in the file.
-LIST	Prints the program on the printer.

FORMAT Parameters

Description

If you do not specify *filename* or *program*, FORMAT prompts for both. If you specify *filename* but not *program*, FORMAT prompts for *program*.

FORMAT is like [FANCY.FORMAT](#), but it handles spacing on each line differently. For example, FORMAT does not add spaces around the = sign.

FORMAT performs the same function as the FORMAT command of the UniVerse Editor.

Examples

To format and print a BASIC program, enter the following:

```
>FORMAT ACCOUNTS ACCOUNTING.LEDGERS -LIST
```

If you enter FORMAT with no qualifiers, FORMAT prompts you:

```
>FORMAT
File name      = FILE1
Spool to printer (Y/N)? N
Record name = A
5 lines long.
```

```
Formatting item "A".
```

```
File name      = FILE1
Record name =
```

After formatting record A, FORMAT prompts you to enter another record ID. Press Return to stop formatting programs.

Here is an unformatted program:

```
FOR X = 1 TO 10
PRINT X
NEXT X
STOP
```

After using the FORMAT command, the program looks like this:

```
FOR X = 1 TO 10
  PRINT X
NEXT X
STOP
```



FORMAT.CONV

Use FORMAT.CONV to change the format of UniVerse files or BASIC object code from one machine's storage format to another. Use this command when you want to transfer UniVerse files or tables between machines with different architectures.

***Note:** This command is not supported on Windows platforms.*

Syntax

FORMAT.CONV [*options*] *pathname*

FORMAT.CONV { *-export* | *-import* | *-convert* [*-vocname name*] [*-force*] | *-drop* | *-undo* } [*-name name*] [*-silent*] *pathname*

Parameters

pathname is the relative or absolute path of the file or table you want to convert. If you are converting a table, you must also specify either the *-export* or the *-import* option.

options must be in lowercase and can be one or more of the following:

Option	Description
<i>-mclass</i>	Converts a file to the format specified by machine class. <i>class</i> can be 0, 1, or 16. See "Description" for a list of machine classes. If you specify multiple options, you must specify the <i>-m</i> option last.
<i>-s</i>	(Silent) Suppresses terminal output.
<i>-u</i>	Converts a file to low-order byte addressing format (big-endian). The <i>-u</i> option assumes a machine class of 0 (zero) for BASIC object code, or a class of 0 or 16 for a UniVerse file. The <i>-u</i> option is the same as <i>-m0</i> .
<i>-v</i>	(Verbose) Displays output. This is the default.

FORMAT.CONV Options

Option	Description
-x	Converts a file to high-order byte addressing format (little-endian). The -x option assumes a machine class of 1. The -x option is the same as -m1.
-6	Converts a 32-bit data file whose format is compatible with Releases 7.3.1 through 9.5.1B to a format compatible with Release 6. -6 converts only files created or resized on a UniVerse Release 9.5.1B or earlier system.
-o	(Old Style) Converts a 32-bit file created or resized on a UniVerse Release 9.5.1C or later system to the older 32-bit file format used on Releases 7.3.1 through 9.5.1B.

FORMAT.CONV Options (Continued)

Use the following options only when you are converting UniVerse tables:

Option	Description
-export	Generates files containing commands used to convert and reconstitute tables transferred to a target schema.
-import	Converts imported tables to the storage format of the current machine. This option executes the commands in the <i>name</i> .IMPORT file.
-convert	Converts imported tables to the storage format of the current machine. This option does not execute the commands in the <i>name</i> .IMPORT file.
-vocname <i>name</i>	Specifies the UniVerse filename as defined in the VOC file if it differs from the operating system filename.
-force	When used with the -convert option, forces FORMAT.CONV to bypass the SQL catalog check.
-drop	Drops tables from the source schema after they have been exported to a target schema.

FORMAT.CONV Options

Option	Description
<code>-undo</code>	Restores imported and converted tables to the state they were in before they were converted with the <code>-import</code> option.
<code>-name name</code>	Specifies the first part of the name of the four text files generated when you use the <code>export</code> option. If you do not specify <i>name</i> , EXPORTEDDDL is used. You cannot use the <code>-name</code> option with the <code>-convert</code> option.
<code>-silent</code>	Suppresses terminal output. Use only with the <code>-export</code> or <code>-convert</code> options.
FORMAT.CONV Options (Continued)	

Description

Use the first syntax for converting the format of UniVerse files. Use the second syntax when you are exporting and importing UniVerse tables either from one system to another or from one schema to another on the same system.

UniVerse Files

If you do not specify `-m`, `-u`, or `-x`, FORMAT.CONV converts the file to the format of the current machine.

FORMAT.CONV changes the file's physical format by reversing the byte-ordering. It does not change the logical contents of files. You can convert the file and transport it to the target machine, or you can transport the file and convert it on the target machine.

The file formats are different on Release 6 and later systems; therefore, the formats are not backward compatible. To use data files created on a later release on a system using an earlier release, you must first convert the files using the `-6` option, then resize the files to convert them to a format compatible with the current system.



Note: The `-6` option converts only data files to Release 6 format. Because it does not convert object code, you need to recompile any BASIC programs created on a later release.

The following table lists hardware platforms supported for Release 10.1 of UniVerse. Use this table to identify your machine class. The UniVerse version code is on your installation tape.

Machine Class	Supported Hardware Platform	UniVerse Version Code
0	IBM RS/6000	uv-029, uv-093, uv-111, uv-301
	Siemens Pyramid	uv-101
	Silicon Graphics	uv-087, uv-089, uv-307
1	Compaq AlphaServer	uv-006, uv-309
	NCR System 3000	uv-028, uv-305
	SCO UnixWare	uv-004
	Sequent Symmetry /ptx	uv-183
	Windows NT	
16	Data General AViiON	uv-120, uv-121, uv-303
	HP 9000 Series 8xx	uv-069, uv-302
	Sun Solaris	uv-024
Platforms Supported in UniVerse 10.0		

If you are running on a UNIX system, you can use the UNIX command *format.conv* to change the file format from a UNIX shell.

UniVerse Tables

When you export and import tables, you use FORMAT.CONV twice. On the source machine, use FORMAT.CONV with the *-export* option to generate a set of files containing commands to be used to convert and reconstitute tables after you transfer them to another schema. After transferring the tables to another schema on either the same or a different machine, use FORMAT.CONV with the *-import* option to convert and reconstitute the transferred tables.

You must be either the owner of the table or a DBA to use the *-export* option. When you execute FORMAT.CONV with *-export*, it examines the table's SICA (security and integrity constraint area) and generates four text files. The default filenames are:

- EXPORTEDDDL.EXPORT
- EXPORTEDDDL.IMPORT

- EXPORTEDDDL.DROP
- EXPORTEDDDL.UNDO

Use the *–name* option to specify an identifier other than EXPORTEDDDL for these files.

On the target machine, execute FORMAT.CONV with the *–import* option. The user who executes this command becomes the owner of the converted table.

FORMAT.CONV with the *–import* option does the following:

- Creates a VOC entry for the table
- Cleans up the table dictionary
- Converts the table to the machine’s storage format
- Creates a new SICA for the table
- Recreates any indexes
- Recreates any triggers
- Grants appropriate permissions
- Recreates any views dependent on the table
- Adds any foreign keys

The procedure for exporting and importing tables from one schema to another, either on the same system or between different systems, is fully described in *UniVerse SQL Administration for DBAs*.

Examples

This example converts the CUSTOMERS file to the format of the current machine:

```
>FORMAT.CONV CUSTOMERS
```

The next two examples convert the ORDERS data file to low-order byte addressing format:

```
>FORMAT.CONV -u ORDERS
>FORMAT.CONV -m0 ORDERS
```

The next example converts the ORDERS file to Release 6 format for a class 1 machine:

```
>FORMAT.CONV -6m1 ORDERS
```

GET.FILE.MAP

Use GET.FILE.MAP in NLS mode to display the name of the map associated with a file.

Syntax

GET.FILE.MAP [DICT] *filename*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
<i>filename</i>	The name of the file, or its path, whose map name you want to display.

GET.FILE.MAP Parameters

Description

GET.FILE.MAP returns the name of the map associated with the specified file. If there is no map name associated with the file, the command gives the name of the default map to be used.

@SYSTEM.RETURN.CODE returns 0 if the command succeeds, or a value less than 0 if there is an error.

Note: If you create a file with NLS mode off and then view it with NLS mode on, GET.FILE.MAP reports the name of the default map in the uvconfig file. If the value in uvconfig is changed, GET.FILE.MAP reports the new name.



Examples

This example displays a map name of NONE for the ORDERS file. The file is stored in NLS internal format, and the data is not mapped.

```
>GET.FILE.MAP ORDERS
Record ID and Data mapping is NONE
```

In the next example, the map named SHIFT-JIS is being used for this file. The file is a hashed file whose map name is retrieved from the NLSDEFFILEMAP parameter in the *uvconfig* file.

```
>GET.FILE.MAP OLD.ACCOUNTS
Record ID and Data mapping is SHIFT-JIS (NLSDEFFILEMAP)
```

The next example displays two map names for the DIVERSION file, both using the defaults from the *uvconfig* file. DIVERSION is a type 1 file: its record IDs are mapped using the JIS-EUC map, and data is mapped using the JIS-EUC+MARKS map.

```
>GET.FILE.MAP DIVERSION
Record ID mapping is JIS-EUC (NLSOSMAP)
Data mapping is      JIS-EUC+MARKS (NLSDEFDIRMAP)
```

GET.LIST

Use GET.LIST to activate a saved select list.

Syntax

GET.LIST [[*filename*] *listname*] [**TO** *list#*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>filename</i>	The name of a file in which the list is saved. The &SAVEDLISTS& file is the default.
<i>listname</i>	The name of the list to be recalled. If you do not include <i>listname</i> , GET.LIST activates the list called &TEMP <i>port#</i> &, where <i>port#</i> is the one- or two-digit identifier of the terminal you are logged in on.
TO <i>list#</i>	The select list number to assign the list to. If you do not use a TO clause, GET.LIST uses select list 0.

GET.LIST Parameters

Description

Activating a select list makes it available to BASIC READNEXT statement, control level commands, data management commands, and Retrieve commands. After you execute GET.LIST, a message confirms that the list is active.

Example

```
>SELECT SUN.MEMBERS WITH LNAME EQ WILLIAMS TO 1

2 record(s) selected to SELECT list #1.
>SAVE.LIST WILLIAMS FROM 1

2 record(s) SAVED to SELECT list "WILLIAMS".
>GET.LIST WILLIAMS
```

2 record(s) selected to SELECT list #0.

>>**LIST SUN.MEMBERS**

MEMBER ID.	FIRST NAME	LAST NAME.	YEAR JOINED	INTERESTS.....
7100	ALICE	WILLIAMS	1984	HANG-GLIDING WINDSURFING
4309	EDGAR	WILLIAMS	1984	FISHING SAILING

2 records listed.

GET.LOCALE

Use GET.LOCALE in NLS mode to display the current locale settings.

Syntax

GET.LOCALE [*category* | ALL]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>category</i>	One of the following locale categories: COLLATE CTYPE MONETARY NUMERIC TIME
ALL	Displays the locale settings (both current and saved, where different) in all categories. ALL is the default.

GET.LOCALE Parameters

Description

GET.LOCALE also displays details of any saved locale that differs from the current one. If locales are not enabled on the system, or if NLS mode is off, GET.LOCALE returns an error.

Examples

```
>GET.LOCALE
```

```
Category:  Current locale  Saved locale if different
```

```
TIME      FR-FRENCH
```

```
NUMERIC   FR-FRENCH
```

```
MONETARY  DE-GERMAN      FR-FRENCH
```

```
CTYPE     FR-FRENCH
```

```
COLLATE   FR-FRENCH
```

```
>GET.LOCALE MONETARY
```

```
Category:  Current locale  Saved locale if different
```

```
MONETARY  DE-GERMAN      FR-FRENCH
```

GET.STACK

Use GET.STACK to retrieve a saved sentence stack from the [&SAVEDLISTS&](#) file and load it into the current sentence stack.

Syntax

GET.STACK [*listname*]

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>listname</i>	The name of a saved sentence stack.

GET.STACK Parameter

Example

```
>GET.STACK MGR.STACK  
Command stack "MGR.STACK" loaded
```

GET.TERM.TYPE

Use GET.TERM.TYPE to display your terminal type, model, and description. If NLS is enabled, GET.TERM.TYPE also displays the map name.

Syntax

GET.TERM.TYPE [**HUSH**]

Parameter

The following table describes each parameter of the syntax.

Parameter	Description
HUSH	Suppresses terminal output. Use HUSH if you want to capture output in a BASIC program.

GET.TERM.TYPE Parameter

Description

The terminal type is set by a [SET.TERM.TYPE](#) command or from the TERM environment variable if no SET.TERM.TYPE command was executed.

Here are some possible terminal types:

Terminal Type	Description
a210	Ampex 210
vp	ADDS Viewpoint
regent25	ADDS Regent 25
vp60	ADDS Viewpoint 60
hz1410	Hazeltine 1400

Terminal Types

Terminal Type	Description
hz1500	Hazeltine 1500
adm5	Lear Siegler ADM 5
tvi925	Televideo 925
tvi955	Televideo 955
wy50	Wyse Technology 50
wy200	Wyse Technology 200
wy300	Wyse Technology 300
dumb	dumb terminal

Terminal Types (Continued)

The terminal type is the name of the file in the directory */usr/lib/terminfo/x* that contains the terminal definition, where *x* is the first character of the code. For example, on UNIX systems the *terminfo* entry for terminal a210 is located in */usr/lib/terminfo/a/a210*. On Windows platforms, if UniVerse is installed in D:\IBM\UV, the entry for terminal a210 is in D:\IBM\UV\terminfo\A\A210.

NLS Mode.

The following additional fields are returned by @SYSTEM.RETURN.CODE:

Field 15 Main terminal map name
Field 16 Auxiliary printer map name

Example

This example shows that the user's terminal is defined as an 80-column by 40-line aiXterm that can display variable-width fonts. The terminal type is *uvterm-v*. The second and third lines show the current terminal display settings as set by the [TERM](#) command.

```
>GET.TERM.TYPE
xterm terminal emulator (small screen 24x80) (xterms)
Width : 80
Depth : 25
Map : ISO8859-1+MARKS
```

GO

Use GO in a paragraph to transfer execution to a subsequent sentence in the paragraph with *label* as its statement label.

Syntax

GO *label* [:]

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>label</i>	The statement label. If you do not specify <i>label</i> , execution continues with the next statement. In a GO statement, the label need not end with a colon.

GO Parameter

Description

The labelled statement must follow the GO statement or the remaining statements will be skipped. You can use the GO statement in an IF statement to exit a loop.

Use GO statements only in paragraphs. It has no function on the command line.

Using Labels

Use labels in a paragraph to uniquely identify a sentence in a paragraph. The syntax of a labelled sentence is as follows:

label: [*sentence*]

A label can be all numbers or it can be any combination of letters, numbers, dollar signs, and periods. It must begin with a letter and end with a colon. A label can be on the same line as a statement, or it can be on a line by itself. You must follow the colon in a label by a space, whether it is on the same line as a sentence or on a line by itself.

You can use labels only in paragraphs. They have no function on the command line.

Example

This example represents a paragraph as it appears in a VOC file:

```
      PAYABLES
0001: PA
0002: IF <<ENTER VENDOR>> = ' ' THEN GO END
0003: LIST PAYABLES <<ENTER VENDOR>> AMT.PD PYMT.DATE
0004: END: DISPLAY Done
```

GRANT.ENCRYPTION.KEY

Use the GRANT.ENCRYPTION.KEY command to grant other users access to the encryption key. When a key is created, only the owner of the key has access. The owner of the key can grant access to other users.

Syntax

GRANT.ENCRYPTION.KEY {PUBLIC | *grantee* {*,grantee...*}}

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
PUBLIC	Grants access to the encryption key to all users on the system.
<i>grantee</i>	Grants access to the encryption key to the <i>grantee</i> you specify. <i>grantee</i> can be a user name, or a group name. If you specify a group name, prefix the name with an asterisk (“*”). When you specify a group name, UniVerse grants access to all users belonging to the group. On Windows platforms, a group name can be a local group or a global group (specified in the form of *Domain\global-group). A user can also be a domain user, specified in the form of Domain\user. In the case of “\” appearing in a group or user name, you should use quotation marks to enclose the name. Grantees cannot grant access to the encryption key to other users. <i>Note: To grant access to global users or groups, you must log on as a domain user to creat keys and perform the GRANT operation.</i>

GRANT.ENCRYPTION.KEY Parameters

Description

You must grant access to an encryption key even if it does not have password protection if you want other users to use the key. On the other hand, even if you have the correct password for the key, you cannot access it without being granted access.

GROUP.STAT

Use GROUP.STAT to produce a summary of the record distribution for a hashed file. This command analyzes the structure of groups within a file. It produces a record distribution summary that can help you determine whether the current file structure is the best one.

Syntax

GROUP.STAT [**DICT**] *filename* [**NO.PAGE**] [**LPTR**]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
<i>filename</i>	The name of the file.
NO.PAGE	Suppresses automatic paging.
LPTR	Sends output to the printer.

GROUP.STAT Parameters

Description

If you do not include *filename* and a select list is active, GROUP.STAT analyzes the files specified by the list.

If you specify a nonhashed file, an error message appears.

Use [GROUP.STAT.DETAIL](#) for more information about a file.

Example

GROUP.STAT produces a summary that look like this:

```
>GROUP.STAT SUN.MEMBERS
Type description= Hashed, keys end in numbers.
Bytes   Records      File= SUN.MEMBERS  Modulo= 2   Sep= 2   Type= 2
  668        7 >>>>>>
  572        6 >>>>>>
=====
1240      13 Totals
  620        6 Averages per group
   48         0 Standard deviation from average
   7.7      0.0 Percent std dev from average
```

GROUP.STAT.DETAIL

Use GROUP.STAT.DETAIL when you want more detailed information about hashed files than [GROUP.STAT](#) provides. This command analyzes the structure of groups within the file. It displays the record ID and number of bytes for each record in each group, that can help you determine whether the current file structure is the best one for this file.

Syntax

GROUP.STAT.DETAIL [DICT] [*filename*] [NO.PAGE] [LPTR]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
<i>filename</i>	The name of the file.
NO.PAGE	Suppresses automatic paging.
LPTR	Sends output to the printer.

GROUP.STAT.DETAIL Parameters

Description

If you do not include *filename* and a select list is active, GROUP.STAT.DETAIL analyzes the filenames specified by the list.

If you specify a nonhashed file, an error message appears.

Example

GROUP.STAT.DETAIL produces a summary that looks like this:

```
>GROUP.STAT.DETAIL SUN.MEMBERS
Type description= Hashed, keys end in numbers.
Bytes Record.id File= SUN.MEMBERS Modulo= 2 Sep= 2 Type= 2
  88 2342
  92 3452
 112 4102
 100 4108
  92 5390
  80 6100
 104 7100
-----
 668 Bytes      7 Records in Group 1

  88 4309
 100 4439
  92 5205
  92 6203
  80 6783
 120 7505
-----
 572 Bytes      6 Records in Group 2

Bytes Records
1240    13 Totals
 620     6 Averages per group
  48     0 Standard deviation from average
 7.7    0.0 Percent std dev from average
```

HASH.AID

Use HASH.AID to experiment with the file type, modulo, and separation for a hashed file and produce a summary of the new record distribution.

Syntax

HASH.AID [DICT] [*filename*] [*type*] [*modulo*] [*separation*] [NO.PAGE]
[LPTR]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
<i>filename</i>	The name of the file.
<i>type</i>	A number from 2 through 18, specifying one of the file types, or *, specifying the existing file type. You can also specify <i>type</i> as a control variable for a loop. See “Description” for details.
<i>modulo</i>	An integer from 1 through 8,388,608 defining the number of groups in the file, or *, specifying the existing modulo. You can also specify <i>modulo</i> as a control variable for a loop. See “Description” for details.
<i>separation</i>	An integer from 1 through 8,388,608 that specifies the group buffer size in 512-byte blocks, or *, specifying the existing separation. You can also specify <i>separation</i> as a control variable for a loop. See “Description” for details.
NO.PAGE	Suppresses automatic paging.
LPTR	Sends output to the printer.

HASH.AID Parameters

Description

If you do not include *filename*, *type*, or *modulo* in the command line, HASH.AID prompts for them. You can enter the value for *type*, *modulo*, and *separation* as control variables for a loop. When you do that, enter the value as follows:

begin,end,increment

begin is the first number to be tried, *end* is the last, and *increment* is an integer. For *type*, the *begin* and *end* numbers must be from 2 through 18. For example, if you enter **HASH.AID SUN.MEMBER 2,3,1 ***, HASH.AID produces two summaries, one using type 2 and the current modulo and one using type 3 and the current modulo.

To keep the existing file type, modulo, or separation, enter ***** instead of a number as the qualifier in the command line. If you are using the prompting method, enter ***** when the prompt for the qualifier you do not want to change appears.

If you specify a nonhashed file, an error message appears.

HASH.AID displays a FILE.STAT of the file being tested and also saves a summary of this information in a file called HASH.AID.FILE. To display this summary, enter the following:

>LIST HASH.AID.FILE WITH FILE.NAME = *filename*

Example

This is the report HASH.AID displays on the terminal:

```
>HASH.AID SUN.MEMBER 3 3,7,2 4 NO.PAGE
File name                               = SUN.MEMBER
File type, modulo, and separation       = 3, 3, 4
Number of records                       = 13
Number of physical bytes                = 8192
Number of data bytes                    = 1240

Average number of records per group    = 4.3333
Minimum number of bytes per group      = 172
Maximum number of bytes per group      = 688
Average number of bytes per group      = 413.3333
Minimum number of records in a group   = 2
Maximum number of records in a group   = 7

Average number of bytes per record     = 95.3846
Minimum number of bytes in a record    = 80
Maximum number of bytes in a record    = 120
```


Average number of fields per record = 10.0000
 Minimum number of fields per record = 10
 Maximum number of fields per record = 10

Groups	25%	50%	75%	100%	125%	150%	175%	200% full
	2	1	0	0	0	0	0	0

File name = SUN.MEMBER
 File type, modulo, and separation = 3, 5, 4
 Number of records = 13
 Number of physical bytes = 12288
 Number of data bytes = 1240

Average number of records per group = 2.6000
 Minimum number of bytes per group = 188
 Maximum number of bytes per group = 488
 Average number of bytes per group = 248.0000
 Minimum number of records in a group = 2
 Maximum number of records in a group = 5

Average number of bytes per record = 95.3846
 Minimum number of bytes in a record = 80
 Maximum number of bytes in a record = 120

Average number of fields per record = 10.0000
 Minimum number of fields per record = 10
 Maximum number of fields per record = 10

Groups	25%	50%	75%	100%	125%	150%	175%	200% full
	5	0	0	0	0	0	0	0

File name = SUN.MEMBER
 File type, modulo, and separation = 3, 7, 4
 Number of records = 13
 Number of physical bytes = 16384
 Number of data bytes = 1240

Average number of records per group = 1.8571
 Minimum number of bytes per group = 100
 Maximum number of bytes per group = 404
 Average number of bytes per group = 177.1429
 Minimum number of records in a group = 1
 Maximum number of records in a group = 4

Average number of bytes per record = 95.3846
 Minimum number of bytes in a record = 80
 Maximum number of bytes in a record = 120

Average number of fields per record = 10.0000
 Minimum number of fields per record = 10
 Maximum number of fields per record = 10

Groups	25%	50%	75%	100%	125%	150%	175%	200% full
	7	0	0	0	0	0	0	0

HASH.HELP

Use HASH.HELP to determine the most efficient file structure for a hashed file. HASH.HELP analyzes the file type, modulo, and separation of an existing file by examining every record ID and record in the file. It determines which record type is most common throughout the file and recommends a file type and modulo.

Syntax

HASH.HELP [DICT][*filename*][NO.PAGE][LPTR]
[PERCENT.GROWTH *n*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
<i>filename</i>	The name of the file.
NO.PAGE	Suppresses automatic paging.
LPTR	Sends output to the printer.
PERCENT.GROWTH	Specifies that the recommendation should be based on <i>n</i> % more records than exist currently in the file. HASH.HELP assumes that the distribution of the additional records is the same as the existing ones.

HASH.HELP Parameters

Description

The HASH.HELP recommendation is based on the averages of the existing records. The modulo it recommends is the minimum modulo and may be smaller than needed. Use the PERCENT.GROWTH keyword to size the file for anticipated growth.

After using HASH.HELP, use [HASH.TEST](#) to experiment with the recommended numbers.

Use [HASH.HELP.DETAIL](#) to get more information about a file.

If you specify a nonhashed file, an error message appears.

Example

HASH.HELP produces a summary that looks like this:

```
>HASH.HELP SUN.MEMBERS
File SUN.MEMBERS Type= 2 Modulo= 2 Sep= 2      11:08:39am 20
Oct 1995 PAGE      1

Of the 13 total keys in this file:

    13 keys were wholly numeric (digits 0 thru 9)
        (Use File Type 2, 6, 10 or 14 for wholly numeric keys)

    0 keys were numeric with separators (as reproduced below)
      0123456789#$$%*+-. /:;_
        (Use File Type 3, 7, 11 or 15 for numeric keys with
separators)

    0 keys were from the 64-character ASCII set reproduced below
      !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN
OPQRSTUVWXYZ[\]^_
        (Use File Type 4, 8, 12 or 16 for 64-character ASCII keys)

    0 keys were from the 256-character ASCII set
        (Use File Type 5, 6, 13 or 17 for 256-character ASCII keys)

The keys in this file are more unique in their right-most eight
bytes.
The smallest modulo you should consider for this file is 3.
The smallest separation you should consider for this file is 1.
The best type to choose for this file is probably type 2.
```

HASH.HELP.DETAIL

Use HASH.HELP.DETAIL to determine the most efficient file structure for a hashed file. [HASH.HELP](#) examines every record ID and record in the file. It determines which record type is most common throughout the file and recommends a file type and modulo.

Syntax

HASH.HELP.DETAIL [DICT] [*filename*] [NO.PAGE] [LPTR]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
<i>filename</i>	The name of the file.
NO.PAGE	Suppresses automatic paging.
LPTR	Sends output to the printer.

HASH.HELP.DETAIL Parameters

Description

In addition to the key and record information that HASH.HELP provides, HASH.HELP.DETAIL also provides the smallest, largest, and average record ID and record data sizes.

The recommendation is based on the averages of the existing records. The recommended modulo is the minimum modulo and may be smaller than needed. To get an alternate recommendation, use HASH.HELP with the PERCENT.GROWTH keyword.

After using HASH.HELP.DETAIL, use [HASH.TEST](#) to experiment with the recommended numbers.

If you specify a nonhashed file, an error message appears.

Example

HASH.HELP.DETAIL produces a summary that looks like this:

```
>HASH.HELP.DETAIL SUN.MEMBERS
File SUN.MEMBERS  Type= 2  Modulo= 2  Sep= 2          11:08:43am  20 Oct 1995
PAGE          1

      13  total keys in file
      52  total bytes in all keys
       4  smallest key
       4  largest key
    4.00  average bytes per key

      13  total records in file
     1240 total bytes in all records
       80 smallest record
      120 largest record
    95.38 average bytes per record

       4  average records per group  (optimized)
     413  average bytes per group  (optimized)

File SUN.MEMBERS  Type= 2  Modulo= 2  Sep= 2          11:08:43am  20 Oct 1995
PAGE          2

Of the 13 total keys in this file:

    13  keys were wholly numeric (digits 0 thru 9)
        (Use File Type 2, 6, 10 or 14 for wholly numeric keys)

     0  keys were numeric with separators (as reproduced below)
    0123456789#$$%*+-./:;_
        (Use File Type 3, 7, 11 or 15 for numeric keys with separators)

     0  keys were from the 64-character ASCII set reproduced below
    !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNQRSTUUVWXYZ[ \ ] ^ _ '
        (Use File Type 4, 8, 12 or 16 for 64-character ASCII keys)

     0  keys were from the 256-character ASCII set
        (Use File Type 5, 6, 13 or 17 for 256-character ASCII keys)

The keys in this file are more unique in their right-most eight bytes.
The smallest modulo you should consider for this file is 3.
The smallest separation you should consider for this file is 1.
The best type to choose for this file is probably type 2.
```

HASH.TEST

Use HASH.TEST to experiment with a file type, modulo, and separation recommended by [HASH.HELP](#) or [HASH.HELP.DETAIL](#) and produce a summary of the new record distribution.

Syntax

HASH.TEST [DICT] [*filename*] [*type*] [*modulo*] [*separation*] [NO.PAGE]
[LPTR]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
<i>filename</i>	The name of the file.
<i>type</i>	A number from 2 through 18, specifying one of the file types, or *, specifying the existing file type. You can also specify <i>type</i> as a control variable for a loop. See “Description” for details.
<i>modulo</i>	An integer from 1 through 8,388,608 defining the number of groups in the file, or *, specifying the existing modulo. You can also specify <i>modulo</i> as a control variable for a loop. See “Description” for details.
<i>separation</i>	An integer from 1 through 8,388,608 that specifies the group buffer size in 512-byte blocks, or *, specifying the existing separation. You can also specify <i>separation</i> as a control variable for a loop. See “Description” for details.
NO.PAGE	Suppresses automatic paging.
LPTR	Sends output to the printer.

HASH.TEST Parameters

Description

If you do not include the file name, type, or modulo in the command line, you are prompted for the missing information.

To keep the existing file type, modulo, or separation, enter ***** instead of a number as the qualifier in the command line or at the prompt.

You can enter the value for type, modulo, and separation as control variables for a loop. When you do that, enter the value as follows:

begin,end,increment

begin is the first number to be tried, *end* is the last, and *increment* is an integer. For *type*, the *begin* and *end* numbers must be from 2 through 9. For example, if you enter **HASH.TEST SUN.MEMBER 2,3,1 ***, HASH.TEST produces two summaries, one using type 2 and the current modulo and one using type 3 and the current modulo.

If you specify a nonhashed file, an error message appears.

Use [HASH.TEST.DETAIL](#) to generate additional information about a file.

Example

HASH.TEST produces a summary that looks like this:

```
>HASH.TEST SUN.MEMBERS
File type           = 2
Modulo              = 5
Separation          = 1
Type description= Hashed, keys end in numbers.
  Bytes  Records      File= SUN.MEMBERS  Modulo= 5  Sep= 1  Type= 2
    488      5  >>>>
      0      0
    292      3  >>>
    272      3  >>>
    188      2  >>
=====  =====
    1240     13  Totals
     248      2  Averages per group
     158      1  Standard deviation from average
    63.7   38.5  Percent std dev from average
```

HASH.TEST.DETAIL

Use HASH.TEST.DETAIL to experiment with a file type, modulo, and separation recommended by [HASH.HELP](#) or [HASH.HELP.DETAIL](#) and produce a detailed summary of the new record distribution, including the size of each group.

Syntax

HASH.TEST.DETAIL [DICT] [*filename*] [*type*] [*modulo*]
[*separation*] [NO.PAGE] [LPTR]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
<i>filename</i>	The name of the file.
<i>type</i>	A number from 2 through 18, specifying one of the file types, or *, specifying the existing file type. You can also specify <i>type</i> as a control variable for a loop. See “Description” for details.
<i>modulo</i>	An integer from 1 through 8,388,608 defining the number of groups in the file, or *, specifying the existing modulo. You can also specify <i>modulo</i> as a control variable for a loop. See “Description” for details.
<i>separation</i>	An integer from 1 through 8,388,608 that specifies the group buffer size in 512-byte blocks, or *, specifying the existing separation. You can also specify <i>separation</i> as a control variable for a loop. See “Description” for details.
NO.PAGE	Suppresses automatic paging.
LPTR	Sends output to the printer.

HASH.TEST.DETAIL Parameters

Description

If you do not include the file name, type, or modulo in the command line, you are prompted for the missing information.

To keep the existing file type, modulo, or separation, enter ***** instead of a number as the qualifier in the command line or at the prompt.

You can enter the value for type, modulo, and separation as control variables for a loop. When you do that, enter the value as follows:

begin,end,increment

begin is the first number to be tried, *end* is the last, and *increment* is an integer. For *type*, the *begin* and *end* numbers must be from 2 through 9. For example, if you enter **HASH.TEST.DETAIL SUN.MEMBER 2,3,1***, HASH.TEST.DETAIL produces two summaries, one using type 2 and the current modulo and one using type 3 and the current modulo.

If you specify a nonhashed file, an error message appears.

Example

HASH.TEST.DETAIL produces a summary that looks like this:

```
>HASH.TEST.DETAIL SUN.MEMBERS
File type           = 2
Modulo              = 5
Separation          = 1
Type description= Hashed, keys end in numbers.
  Bytes  Records      File= SUN.MEMBERS  Modulo= 5  Sep= 1  Type= 2
    120   7505
     92   5205
    104   7100
     80   6100
     92   5390
-----
    488 Bytes      5 Records in Group 1
-----
     0 Bytes      0 Records in Group 2
-----
    112   4102
     92   3452
     88   2342
-----
    292 Bytes      3 Records in Group 3
```

80	6783	
92	6203	
100	4108	

272	Bytes	3 Records in Group 4
100	4439	
88	4309	

188	Bytes	2 Records in Group 5
Bytes	Records	
1240	13	Totals
248	2	Averages per group
158	1	Standard deviation from average
63.7	38.5	Percent std dev from average

HELP

Use HELP to display a description of a UniVerse command or keyword, a conversion code, a BASIC statement or function, a UniVerse SQL statement, or a BASIC SQL Client Interface function.

Syntax

HELP [BASIC | CONV | PICK | SQL | BCI | *helpfile*] [*item* | *]

HELP [*item*] FROM *helpfile*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
BASIC	Lists help about the BASIC command. Followed by the name of a BASIC statement or function, the BASIC keyword specifies help for a BASIC statement or function.
CONV	Specifies help for conversion codes. If you do not specify <i>item</i> , CONV lists all conversion codes. If you specify a conversion code as <i>item</i> , help for that code is listed.
PICK	Specifies help for PICK, REALITY, and IN2 flavor accounts. If you do not specify <i>item</i> , PICK lists commands with PICK-flavor-specific differences. If you specify a command as <i>item</i> , help for that command is listed.
SQL	Specifies help for SQL statements. If you do not specify <i>item</i> , SQL lists all SQL statements. If you specify an SQL statement as <i>item</i> , help for that statement is listed.
BCI	Specifies help for BASIC SQL Client Interface functions. If you do not specify <i>item</i> , BCI lists all SQL Client Interface functions. If you specify a function as <i>item</i> , help for that function is listed.

HELP Parameters

Parameter	Description
<i>helpfile</i>	Specifies one of the system help files: BASIC.HELP BCI.HELP CONV.HELP PICK.HELP SQL.HELP SYS.HELP
<i>item</i>	The name of a UniVerse command or keyword, a BASIC statement or function, a conversion code, an SQL statement, or a UniVerse BASIC SQL Client Interface function.
*	Lists all items specified by the preceding keyword. If you do not include a keyword in the command line, * lists help items for UniVerse commands and keywords.
FROM <i>helpfile</i>	Specifies one of the system help files.

HELP Parameters (Continued)

If you include *item* in the command line, UniVerse displays the description of that command, statement, or keyword. If you enter HELP with no options, you get a list of UniVerse commands for which there is online help.

Description

You can also get help by entering **?item**. If field 1 in a record in the VOC file contains a description, **?item** displays it.

Some commands, statements, and keywords have more than one help entry. To display more help, select the End Help option at the bottom of the help screen.

UniVerse supplies the following help files: BASIC.HELP, BCI.HELP, CONV.HELP, PICK.HELP, SQL.HELP, and SYS.HELP. They contain all the descriptions for system commands and functions, conversion codes, and BASIC statements and functions. A UniVerse Administrator can create a USER.HELP file to include descriptions of special commands, sentences, programs, and files that are unique to your system. To create the USER.HELP file, begin by examining the SYS.HELP file. Then use [CREATE.FILE](#) to create a file called USER.HELP and follow the format of SYS.HELP for your help descriptions.

Examples

This example lists UniVerse commands and keywords for which there is online help:

```
>HELP
```

The next example displays help for the COPY command:

```
>HELP COPY
```

The next example displays help for the BASIC LOCATE statement:

```
>HELP BASIC LOCATE
```

The next example displays help for the *Tfile* conversion:

```
>HELP CONV Tfile
```

HUSH

Use HUSH to suppress all text normally sent to the screen during processing.

Syntax

HUSH [ON | OFF]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
ON	Suppresses display of all terminal output, including error messages and prompts for input.
OFF	Enables display of terminal output.

HUSH Parameters

Description

You might use this command when you are transmitting information over phone lines or when you are sending data to a hard-copy terminal. Both these situations result in slower transmission speeds. The unnecessary data display makes the task even slower.

HUSH acts as a toggle. If you use HUSH without a qualifier, it changes the current state.

Do not use this command to shut off output display unless you are sure it is unnecessary. When you use HUSH ON, all output display is suppressed, including error messages and requests for information.

IAM

Use IAM to change the name of your account directory.

Syntax

IAM *account.dir*

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>account.dir</i>	The account directory name you want to use.

IAM Parameter

Description

Some commands require an account directory name. [CATALOG](#), for example, uses the account directory name as part of the name of a cataloged program when you use normal cataloging.

When you enter the UniVerse environment, UniVerse sets your account directory name to the current directory name. When you use the [LOGTO](#) command, your account directory name is set to the name of the directory of the account you log to. Use IAM to change the account directory name so that commands like CATALOG use the name you specify.

Use [WHO](#) to see what the current account directory name is. It is helpful to use WHO before and after IAM.

IAM does not change your working directory. Use [CHDIR](#) or LOGTO to change your working directory.

Example

This example uses WHO to display the user's terminal number (38), account directory (*uv*), and login name (*ken*). The user changes the account directory name to *mark* and uses WHO again to confirm the change.

```
>WHO
38 uv From ken
>IAM mark
>WHO
38 mark From ken
```

UniVerse uses *mark* as the account directory name when a command requires it. For instance, if you use the CATALOG command to catalog programs normally, it uses *mark* instead of *uv* as the account directory name.

IF

Use IF in a paragraph to change the operation sequence. The IF statement introduces a test whose results determine the next action. You must use an IF statement in a loop to end the loop (see “Example”). IF statements have no function on the command line.

Syntax

IF *expression* THEN *sentence*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>expression</i>	The test to be performed. Its syntax is <i>value1 operator value2</i> .
<i>value1</i>	An inline prompt specification, @variable, or constant. In a loop, use control option A in the inline prompting expression (see <<...>>) to issue the prompt every time the loop executes the IF statement, otherwise the loop issues the prompt only once.
<i>operator</i>	Any relational operator. For a complete list, see Chapter 2, “UniVerse Keywords.”
<i>value2</i>	Must be a constant, an @variable, or an inline prompt.
<i>sentence</i>	The sentence to execute if <i>expression</i> evaluates to TRUE.

IF Parameters

Example

This example shows a paragraph as it appears in the VOC file:

```
001: PA
002: LOOP
003: IF <<A, End of Month (Y or N)>> = 'Y' THEN GO MONTH
004: LIST <<A, Enter filename>> VENDOR AMT.PAID PYMT.DATE AMT.DUE
005: REPEAT
006: MONTH: EOM.REPORT
007: END: DISPLAY Done
```

INITIALIZE.CATALOG

Use INITIALIZE.CATALOG to delete the system catalog space and reinitialize it. You must be a UniVerse Administrator logged in to the UV account to use this command.

Syntax

INITIALIZE.CATALOG [-R]

Parameter

The following table describes each parameter of the syntax.

Parameter	Description
-R	Reinitializes the system catalog space and overwrites standard UniVerse-supplied programs. Although this option does not delete the system catalog space, use caution when using -R.

INITIALIZE.CATALOG Parameters

Description

The impact of INITIALIZE.CATALOG can be catastrophic. You should be absolutely sure you want to initialize the catalog space. If you have any doubts whatsoever about this, do not press **Y** at the `Continue Initialization` prompt.

Example

This example reinitializes the system catalog space:

```
>INITIALIZE.CATALOG
```

```
*****
*                                     WARNING
*
*      You are about to destroy the system Catalog space. This
*      may have a significant effect upon the user community.
*
*****
Continue Initialization of the Catalog space? (Y/N) <Return>

Initialization of CATALOG space completed.
```

INITIALIZE.DEMO

Use INITIALIZE.DEMO to initialize three demonstration files in your account.

Syntax

INITIALIZE.DEMO

Description

UniVerse supplies the sample files CUSTOMERS, INVENTORY, and ORDERS. INITIALIZE.DEMO updates the VOC file, then verifies that files with these names do not exist. If a file with one of these names exists, INITIALIZE.DEMO checks to see if it is an old demonstration file. If it is, INITIALIZE.DEMO clears the file without prompting you and loads new records into the file. If the file is not an old demonstration file, INITIALIZE.DEMO asks if you want to overwrite it. This command also checks for the existence of an ACCOUNTS file. If it exists, INITIALIZE.DEMO asks if you want to delete it.

JOBS

Use JOBS to list currently running phantom processes that were started from your current UniVerse process.

Syntax

JOBS

Description

Phantom processes are processes that run in the background. You start them with the PHANTOM command.

There is a systemwide limit on the number of processes you can initiate. If you can start no more processes when you issue a PHANTOM command, a message tells you there are no free phantoms.

For more information about phantom processes, see the [PHANTOM](#) command.

Example

This example shows that process 625 is running:

```
>JOBS  
[625] Running : LIST SALES  
>
```

LIMIT

Use LIMIT to set the maximum size of memory storage for a user’s active BASIC routines.

Syntax

LIMIT [PROGRAMSIZE *size*]

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>size</i>	Expressed as a number of 1024-byte units. Zero (0) specifies unlimited space.

LIMIT Parameters

Description

If you do not specify PROGRAMSIZE, LIMIT displays the current value.

When a routine that exceeds *size* is loaded into a user’s memory space, UniVerse tries to unload the least recently used programs to bring the total usage below *size*.

LIST

Use LIST to display selected data from a file. You can sort and format the output of LIST in many ways.

Syntax

```
LIST [ DICT | USING [ DICT ] dictname ] filename [ records | FROM n ]  
[ selection ] [ output.limiter ] [ sort ] [ output ] [ report.qualifiers ] [ TOXML  
[ ELEMENTS ] [ WITHDTD ] [ XMLMAPPING mapping_file ] ]
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Lists records in the file dictionary of <i>filename</i> . If you do not specify DICT, records in the data file are listed.
USING [DICT] <i>dictname</i>	If DICT is not specified, uses <i>dictname</i> as the dictionary of <i>filename</i> . If DICT is specified, the dictionary of <i>dictname</i> is used as the dictionary of <i>filename</i> .
<i>filename</i>	The file whose records you want to list. You can specify <i>filename</i> anywhere in the sentence. LIST uses the first word in the sentence that has a file descriptor in the VOC file as the file name.
<i>records</i>	Specifies the records to include in the list. You can specify as many record IDs as you want, separated by spaces. Enclose record IDs in single quotation marks to ensure that RetriVe does not interpret them as field names or keywords.
FROM <i>n</i>	Specifies the records whose record IDs are stored in select list <i>n</i> .

LIST Parameters

Parameter	Description								
<i>selection</i>	A selection expression specifying the conditions that data in a record must meet for the record to be selected. A selection expression begins with the keyword WITH. For syntax details, see the WITH keyword in Chapter 2, “UniVerse Keywords.”								
<i>output.limiter</i>	An expression specifying the conditions that values in multivalued fields must meet for those values to be output. An output-limiting expression begins with the keyword WHEN. For syntax details, see the WHEN keyword in Chapter 2, “UniVerse Keywords.”								
<i>sort</i>	A sort expression that specifies the type of sort and the field to sort on. The syntax is as follows: <table> <tr> <td><i>BY field</i></td><td>Sort on <i>field</i> in ascending order.</td></tr> <tr> <td><i>BY.DSND field</i></td><td>Sort on <i>field</i> in descending order.</td></tr> <tr> <td><i>BY.EXP field</i></td><td>Explode multivalues in <i>field</i> and <i>sort</i> in ascending order.</td></tr> <tr> <td><i>BY.EXP field</i></td><td>Explode multivalues in <i>field</i> and <i>sort</i> in descending order.</td></tr> </table> For more information about sort expressions, see Chapter 2, “UniVerse Keywords.” When NLS locales are enabled, LIST uses the Collate convention of the current locale to determine the collating order for any existing sort expression. If an index has its own Collate convention defined, LIST uses it instead of the current locale definition. For more information, see the <i>UniVerse NLS Guide</i>.	<i>BY field</i>	Sort on <i>field</i> in ascending order.	<i>BY.DSND field</i>	Sort on <i>field</i> in descending order.	<i>BY.EXP field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in ascending order.	<i>BY.EXP field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in descending order.
<i>BY field</i>	Sort on <i>field</i> in ascending order.								
<i>BY.DSND field</i>	Sort on <i>field</i> in descending order.								
<i>BY.EXP field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in ascending order.								
<i>BY.EXP field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in descending order.								

LIST Parameters (Continued)

Parameter	Description																								
<i>output</i>	Specifies the fields whose data you want to list. The fields must be defined in the file dictionary. If you are listing dictionary records, the fields must be defined in the DICT.DICT file. You can precede a field name with one field modifier. Field modifiers are: <table><tr><td>AVG</td><td>ENUM</td><td>PCT</td></tr><tr><td>BREAK.ON</td><td>MAX</td><td>TOTAL</td></tr><tr><td>BREAK.SUP</td><td>MIN</td><td>TRANSPORT</td></tr><tr><td>CALC</td><td></td><td></td></tr></table> You can follow a field name with one or more field qualifiers. Field qualifiers are: <table><tr><td>AS</td><td>COL.HDG</td><td>FMT</td></tr><tr><td>ASSOC</td><td>CONV</td><td>MULTI.VALUE</td></tr><tr><td>ASSOC.WITH</td><td>DISPLAY.LIKE</td><td>SINGLE.VALUE</td></tr></table> For information about these keywords and their synonyms, see Chapter 2, “UniVerse Keywords.” If you do not specify <i>output</i>, fields specified in the @ phrase are listed. If there is no @ phrase in the file dictionary, only record IDs are listed.	AVG	ENUM	PCT	BREAK.ON	MAX	TOTAL	BREAK.SUP	MIN	TRANSPORT	CALC			AS	COL.HDG	FMT	ASSOC	CONV	MULTI.VALUE	ASSOC.WITH	DISPLAY.LIKE	SINGLE.VALUE			
AVG	ENUM	PCT																							
BREAK.ON	MAX	TOTAL																							
BREAK.SUP	MIN	TRANSPORT																							
CALC																									
AS	COL.HDG	FMT																							
ASSOC	CONV	MULTI.VALUE																							
ASSOC.WITH	DISPLAY.LIKE	SINGLE.VALUE																							
<i>report.qualifiers</i>	One or more of the following keywords: <table><tr><td>COL.HDR.SUPP</td><td>FIRST</td><td>ID.ONLY</td><td>ONLY</td></tr><tr><td>COL.SPCS</td><td>FOOTING</td><td>ID.SUP</td><td>SAMPLE</td></tr><tr><td>COL.SUP</td><td>GRAND.TOTAL</td><td>LPTR</td><td>SAMPLED</td></tr><tr><td>COUNT.SUP</td><td>HDR.SUP</td><td>MARGIN</td><td>SUPP</td></tr><tr><td>DBL.SPC</td><td>HEADING</td><td>NO.SPLIT</td><td>VERT</td></tr><tr><td>DET.SUP</td><td></td><td>NOPAGE</td><td></td></tr></table> These keywords modify the report format. For information about them and their synonyms, see Chapter 2, “UniVerse Keywords.” If you do not specify any report qualifiers, LIST produces a report with: - n UniVerse default heading and footing - n Single spacing between records - n Column headings on every page - n No control breaks - n Paged output to the screen	COL.HDR.SUPP	FIRST	ID.ONLY	ONLY	COL.SPCS	FOOTING	ID.SUP	SAMPLE	COL.SUP	GRAND.TOTAL	LPTR	SAMPLED	COUNT.SUP	HDR.SUP	MARGIN	SUPP	DBL.SPC	HEADING	NO.SPLIT	VERT	DET.SUP		NOPAGE	
COL.HDR.SUPP	FIRST	ID.ONLY	ONLY																						
COL.SPCS	FOOTING	ID.SUP	SAMPLE																						
COL.SUP	GRAND.TOTAL	LPTR	SAMPLED																						
COUNT.SUP	HDR.SUP	MARGIN	SUPP																						
DBL.SPC	HEADING	NO.SPLIT	VERT																						
DET.SUP		NOPAGE																							

LIST Parameters (Continued)

Parameter	Description
TOXML	Outputs LIST results in XML format.
ELEMENTS	Outputs results in element-centric format. I
WITHDTD	Output produces a DTD corresponding to the query.
XMLMAPPING <i>mapping_file</i>	Specifies a mapping file containing transformation rules for display. This file must exist in the &XML& file.

LIST Parameters (Continued)

Description

LIST lists the selected records in the order in which they are stored in the file, unless you specify a sort expression.

Specifying Output

In an *output* specification you can specify as many field names as you want, separated by spaces. You can specify field names explicitly or as a phrase containing field names. You can use field names and phrases together in the same output specification. Data in the specified fields is listed in columns in the order in which field names appear in the sentence or @ phrase.

Using Field Expressions

You can use field expressions in selection expressions, sort expressions, output specifications, and report clauses. A field expression can be a field name with or without field qualifiers, or it can be an EVAL expression. An EVAL expression is an I-type expression specified on the command line, introduced by the keyword EVAL.

Suppressing Record IDs

By default, LIST displays record IDs in the first column of output. Use the [ID.SUP](#) keyword to suppress the display of record IDs.

LIST.DF

Use LIST.DF to list the pathnames and part numbers of part files belonging to a distributed file.

Syntax

LIST.DF *dist.filename*

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>dist.filename</i>	The name of a distributed file.

LIST.DF Parameter

Example

This example lists two part files (PART1 and PART2) belonging to the distributed file DIST.FILE:

```
>LIST.DF DIST.FILE
Part file "/usr/SALES/PART1", Part number = 1.
Part file "/usr/SALES/PART2", Part number = 2.
>
```

LIST.DIFF

Use LIST.DIFF to compare two saved lists and create a third list of elements from list one that are not in list two.

Syntax

LIST.DIFF [*listname1*] [*options*]

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>listname1</i>	The name of a saved list in the &SAVEDLISTS& file. If you do not specify <i>listname1</i> , LIST.DIFF uses the list whose name is an empty string.

LIST.DIFF Parameter

options can be one or more of the following:

Option	Description
LPTR	Sends output to the printer.
CRT	Sends output to the terminal.
NOPAGE	If output goes to the terminal, suppresses automatic paging.
OVERWRITING	Overwrites an existing list in the &SAVEDLISTS& file.
NO.WARN	If output goes to the terminal or printer, suppresses line numbers.
HEX	Lists output in hexadecimal format.

LIST.DIFF Options

Description

The system prompts you to enter the name of a second list by displaying the following prompt:

WITH:

Enter the name of a second list. _

If CRT or LPTR is not specified, the system prompts you for a destination for the result of the LIST.DIFF operation by displaying the following prompt:

TO:

Enter the name of the destination file using one of the following formats:

```
[ dest.listname [ account.name ] ]  
(dest.file) record.ID
```

If you use the first form, LIST.DIFF copies the original list to the destination list, *dest.listname*. If you do not specify *dest.listname*, LIST.DIFF uses an empty string as the list name. If you specify *account.name*, LIST.DIFF creates the new list in the &SAVEDLIST& file of the specified account. If you specify the OVERWRITING option, the new list overwrites any existing list with the same name.

If you use the second form, LIST.DIFF converts the list to a record in *dest.file*, and each element in the list constitutes a field in the record. If the size of the list exceeds 32,267 bytes, the record is truncated.

NLS Mode

When locales are enabled, LIST.DIFF uses the Collate convention of the current locale to determine the collating order for the new list. For more information, see the *UniVerse NLS Guide*.

Example

This example creates and saves two select lists: 1991 and 1992. Comparing the two saved lists, LIST.DIFF creates and saves a third select list, DIFF.92, which contains elements from list 1991 that are not in list 1992.

```
>SELECT SUN.MEMBER WITH YR.JOIN EQ "1991"  
2 record(s) selected to SELECT list #0.  
>>SAVE.LIST 1991  
2 record(s) SAVed to SELECT list "1991".  
>SELECT SUN.MEMBER WITH YR.JOIN >= "1991"  
7 record(s) selected to SELECT list #0.  
>>SAVE.LIST 1992  
7 record(s) SAVed to SELECT list "1992".  
>LIST.DIFF 1992  
WITH: 1991  
TO: DIFF.92  
5 record(s) SAVed to SELECT list "DIFF.92".
```

LIST.ENCRYPTION.FILE

Use the LIST.ENCRYPTION.FILE command to display encryption configuration data, such as the fields that are encrypted, the algorithms used, and so forth. This command also displays the fields for which decryption is currently disabled.

Syntax

LIST.ENCRYPTION.FILE *filename*

LIST.ENCRYPTION.KEY

Use the LIST.ENCRYPTION.KEY command to list the existing keys in the key store.

Syntax

LIST.ENCRYPTION.KEY

Description

You can also list records in the key store using UniVerse Retrieve commands, such as LIST, LIST.ITEM, SORT, SORT.ITEM, and so forth.



***Note:** The name of the key store file is &KEYSTORE&. Although you can view records from this file using UniVerse Retrieve commands, other UniVerse commands, such as DELETE.FILE and CLEAR.FILE will fail. The ED command will only display encrypted data.*

LIST.ENCRYPTION.WALLET

Use the LIST.ENCRYPTION.WALLET command to list the existing encryption wallets keys in the key store.

Syntax

LIST.ENCRYPTION.WALLET

Description

You can also list records in the key store using UniVerse Retrieve commands, such as LIST, LIST.ITEM, SORT, SORT.ITEM, and so forth.



Note: The name of the key store file is &KEYSTORE&. Although you can view records from this file using UniVerse Retrieve commands, other UniVerse commands, such as DELETE.FILE and CLEAR.FILE will fail. The ED command will only display encrypted data.

LIST.FILE.STATS

Use LIST.FILE.STATS to display or print file statistics. You must use the [ACCOUNT.FILE.STATS](#) command to gather the statistics before you can list them with LIST.FILE.STATS.

Syntax

LIST.FILE.STATS [LOCAL] [WIDE] [LPTR]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
LOCAL	Lists statistics in the STAT.FILE file in the local account. If you do not specify LOCAL, the statistics in UNIVERSE.STAT.FILE, in the UV account, are listed.
WIDE	Lists file statistics in 132-column format. If you do not specify WIDE, the report uses 80-column format.
LPTR	Sends output to the printer.

LIST.FILE.STATS Parameters

Description

If you enter LIST.FILE.STATS with no options, the command generates a listing of statistics stored in the UNIVERSE.STAT.FILE file.

Example

The LIST.FILE.STATS command produces a report such as the following:

```
      * * * F I L E      S T A T S      R E C O R D * * *                Page   1

/usr/walter gathered on 01 OCT 1995 at 14:09:21.

                                Record.  File.....
File Name....  Type Mod.  Sep Count.. Size (ext)   25%   50%   75%   >100%

D_&COMO&             18    1    1         1       1536    1    0    0
&COMO&              1         17       34330
D_&DEVICE&           3    1    2        20       3072    0    0    0    1
&DEVICE&            2    3    4        15       8192    2    0    1
D_&ED&               3    1    2         1       2048    1    0    0
&ED&                1         9        757
D_&FILESTATS&        2    3    4        44      12288    0    0    1    2
&FILESTATS         3   17    4         7      36864   17    0    0
D_&HOLD&             3    1    2         1       2048    1    0    0
&HOLD&              1         1        286
D_&PH&               3    1    2         1       2048    1    0    0
&PH&                1         2      61080
D_&SAVEDLISTS&       3    1    2         1       2048    1    0    0
&SAVEDLISTS&       1         33     123803
D_&TEMP&             3    1    2         1       2048    1    0    0
&TEMP&              1         8        199

Press any key to continue...
```

LIST.INDEX

Use LIST.INDEX to display information about a file's secondary indexes or an SQL table's indexes.

Syntax

LIST.INDEX [DICT] [*filename*] [*indexes* | ALL] [STATISTICS | STATS |
DETAIL] [LPTR [*n*]] [NO.PAGE]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file or table dictionary.
<i>filename</i>	The name of a UniVerse file or table. If you do not specify <i>filename</i> , LIST.INDEX prompts for it.
<i>indexes</i>	A list of indexed fields separated by spaces. <i>indexes</i> are the names of fields in <i>filename</i> used as secondary index keys.
ALL	Lists all the secondary indexes in the file.
STATISTICS	Lists more information about the secondary index, including the total number of keys, records per key, and index size. LIST.INDEX scans each index to gather the data, so this process can take a while for large files.
STATS	The same as STATISTICS.
DETAIL	In addition to the information from the STATISTICS keyword, displays the name of each secondary key value, the number of records with that key value, and the bytes used for that key value.
LPTR	Sends output to the printer via logical print channel <i>n</i> . If you do not specify <i>n</i> , 0 is assumed.
NO.PAGE	Suppresses automatic paging.

LIST.INDEX Parameters



Description

If you do not specify either *indexes* or ALL, LIST.INDEX prompts you.

LIST.INDEX displays general information about specified secondary indexes.

General information appears at the top of the report for each index, including the file name, the number and type of indexes present, the update mode of the index (enabled or disabled), and whether the index requires updating. In NLS mode, LIST.INDEX displays the locale and map names for the indexes.

Note: *SQL indexes are like I-type indexes. The name of the SQL index corresponds to an I-descriptor with that name.*

After the general information, LIST.INDEX provides the following information for each index:

Heading	Value	Description
Index name	<i>fieldname</i>	The name of the indexed field or SQL index
Type	D	Data descriptor
	I	I-descriptor
	A	A-descriptor
	S	S-descriptor
	C	A- or S-descriptor with correlative in field 8
Build	SQL	SQL index.
	Required	Index needs to be built
Nulls	Not Reqd	Index does not need to be built
	Yes	Empty strings are indexed
In DICT	No	Empty string are not indexed
	Yes	File dictionary contains corresponding field
S/M	No	File dictionary does not contain corresponding field
	S	Single-valued index
Just	M	Multivalued index
	L	Left or text justification
	R	Right justification

LIST.INDEX Information

Heading	Value	Description
Unique	Y	Field or SQL index has UNIQUE column constraint
	N	Field or SQL index has no UNIQUE column constraint
Field num	<i>field number</i>	If Type is D, A, or S
I-type	<i>I-type expression</i>	If Type is I or SQL
	<i>correlative</i>	If Type is C
NLS Collation Locale	<i>locale.name</i>	The name of the locale associated with the index.

LIST.INDEX Information (Continued)

If you specify the name of a distributed file, LIST.INDEX lists general information about all part files. LIST.INDEX lists further information about an index only if the corresponding fields in all part files have similar indexes. For indexes to be similar, the following elements of the field definitions must be identical:

- Type
- Justification
- S/M (single-valued or multivalued)
- Field number
- Compiled I-type object code

Example

This example displays information about the secondary index for the SORT.FRENCH file with NLS enabled:

```
>LIST.INDEX SORT.FRENCH ALL
Alternate Key Index Summary for file SORT.FRENCH
File..... SORT.FRENCH
Indices..... 1 (0 A-type, 0 C-type, 1 D-type, 0 I-type, 0 SQL, 0 S-type)
Index Updates.. Enabled, No updates pending

Index name      Type  Build   Nulls  In DICT  S/M   Just Unique Field
num/I-type
@ID             D     Not Reqd Yes    Yes      S     L     N     0
              NLS Collation Locale... FR-FRENCH ; * Not Loaded

>
```

LIST.INTER

Use LIST.INTER to compare two saved lists and create a third list of elements from list one that are also found in list two, which are not redundant.

Syntax

LIST.INTER [*listname1*] [*options*]

Parameter

The following table lists the parameter of the syntax.

Parameter	Description
<i>listname1</i>	The name of a saved list in the &SAVEDLISTS& file. If you do not specify <i>listname1</i> , LIST.INTER uses the list whose name is an empty string.

LIST.INTER Parameters

options can be one or more of the following:

Parameter	Description
LPTR	Sends output to the printer.
CRT	Sends output to the terminal.
NOPAGE	If output goes to the terminal, suppresses automatic paging.
OVERWRITING	Overwrites an existing list in the &SAVEDLISTS& file.
NO.WARN	If output goes to the terminal or printer, suppresses line numbers.
HEX	Lists output in hexadecimal format.

LIST.INTER Options

Description

The system prompts you to enter the name of a second list by displaying the following prompt:

WITH:

Enter the name of a second list. **_**

If CRT or LPTR is not specified, the system prompts you for a destination for the result of the LIST.INTER operation, by displaying the following prompt:

TO:

Enter the name of the destination file using one of the following formats:

`[dest.listname [account.name]]`
`(dest.file) record.ID`

If you use the first form, LIST.INTER copies the original list to the destination list, *dest.listname*. If you do not specify *dest.listname*, LIST.INTER creates a list whose name is an empty string. If you specify *account.name*, LIST.INTER creates the new list in the &SAVEDLIST& file of the specified account. If you use the OVERWRITING option, the new list overwrites any existing list with the same name.

If you use the second form, LIST.INTER converts the list to a record in *dest.file*, and each element in the list constitutes a field in the record. If the size of the list exceeds 32,267 bytes, LIST.INTER truncates the record.

NLS Mode

When locales are enabled, LIST.INTER uses the Collate convention of the current locale to determine the collating order for the new list. For more information, see the *UniVerse NLS Guide*.

Example

This example creates and saves two select lists: 1991 and 1992. Comparing the two saved lists, LIST.INTER creates and saves a third select list, INTER.92, which contains elements from list 1991 that are also in list 1992 and that are not redundant.

```
>SELECT SUN.MEMBER WITH YR.JOIN EQ "1991"
2 record(s) selected to SELECT list #0.
>>SAVE.LIST 1991
2 record(s) SAVED to SELECT list "1991".
>SELECT SUN.MEMBER WITH YR.JOIN >= "1991"
7 record(s) selected to SELECT list #0.
>>SAVE.LIST 1992
7 record(s) SAVED to SELECT list "1992".
>LIST.INTER 1991
WITH: 1992
TO: INTER.92
2 record(s) SAVED to SELECT list "INTER.92".
```

LIST.ITEM

Use LIST.ITEM to display a complete listing of selected records.

Syntax

LIST.ITEM [DICT | USING [DICT] *dictname*] *filename* [*records* | FROM *n*]
[*selection*] [*sort*] [*report.qualifiers*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Lists records in the file dictionary of <i>filename</i> . If you do not specify DICT, records in the data file are listed.
USING [DICT] <i>dictname</i>	If DICT is not specified, uses <i>dictname</i> as the dictionary of <i>filename</i> . If DICT is specified, the dictionary of <i>dictname</i> is used as the dictionary of <i>filename</i> .
<i>filename</i>	The file whose records you want to list. You can specify <i>filename</i> anywhere in the sentence. LIST.ITEM uses the first word in the sentence that has a file descriptor in the VOC file as the file name.
<i>records</i>	Specifies the records to include in the list. You can specify as many record IDs as you want, separated by spaces. Enclose record IDs in single quotation marks to ensure that Retrieve does not interpret them as field names or keywords.
FROM <i>n</i>	Specifies the records whose record IDs are stored in select list <i>n</i> .
<i>selection</i>	A selection expression specifying the conditions that data in a record must meet for the record to be selected. A selection expression begins with the keyword WITH. For syntax details, see the WITH keyword in Chapter 2, “UniVerse Keywords.”
<i>sort</i>	A sort expression that specifies the type of sort and the field to sort on. The syntax is as follows:

LIST.ITEM Parameters

Parameter	Description																
	BY <i>field</i> Sort on <i>field</i> in ascending order. BY.DSND <i>field</i> Sort on <i>field</i> in descending order. BY.EXP <i>field</i> Explode multivalues in <i>field</i> and <i>sort</i> in ascending order. BY.EXP.DSND <i>field</i> Explode multivalues in <i>field</i> and <i>sort</i> in descending order. For more information about sort expressions, see Chapter 2, “UniVerse Keywords.” When NLS locales are enabled, LIST.ITEM uses the Collate convention of the current locale to determine the collating order for any existing sort expression. If an index has its own Collate convention defined, LIST.ITEM uses it instead of the current locale definition. For more information, see the <i>UniVerse NLS Guide</i>.																
<i>report.qualifiers</i>	One or more of the following keywords: <table><tr><td>COL.HDR.SUPP</td><td>HDR.SUP</td><td>LPTR</td><td>SAMPLE</td></tr><tr><td>DBL.SPC</td><td>HEADING</td><td>MARGIN</td><td>SAMPLED</td></tr><tr><td>FIRST</td><td>ID.SUP</td><td>NOPAGE</td><td>SUPP</td></tr><tr><td>FOOTING</td><td></td><td></td><td></td></tr></table> These keywords modify the report format. For information about them and their synonyms, see Chapter 2, “UniVerse Keywords.” If you do not specify any report qualifiers, LIST.ITEM produces a report with: - n UniVerse default heading - n Single spacing between records - n Paged output to the screen	COL.HDR.SUPP	HDR.SUP	LPTR	SAMPLE	DBL.SPC	HEADING	MARGIN	SAMPLED	FIRST	ID.SUP	NOPAGE	SUPP	FOOTING			
COL.HDR.SUPP	HDR.SUP	LPTR	SAMPLE														
DBL.SPC	HEADING	MARGIN	SAMPLED														
FIRST	ID.SUP	NOPAGE	SUPP														
FOOTING																	

LIST.ITEM Parameters (Continued)

Description

Retrieve field output specifications are ignored.

LIST.ITEM is like the Pick version of the **COPY** command (using the P or T option), but it lets you specify selection criteria and headings and footings.

LIST.LABEL

Use LIST.LABEL to specify a format suitable for mailing labels and other specialized block listings. The report lists the fields for each record in a block, with headings in the leftmost column.

Syntax

```
LIST.LABEL [ DICT | USING [ DICT ] dictname ] filename
[ records | FROM n ] [ selection ] [ output.limiter ] [ sort ] [ output ]
[ report.qualifiers ]
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Lists records in the file dictionary of <i>filename</i> . If you do not specify DICT, records in the data file are listed.
USING [DICT] <i>dictname</i>	If DICT is not specified, uses <i>dictname</i> as the dictionary of <i>filename</i> . If DICT is specified, the dictionary of <i>dictname</i> is used as the dictionary of <i>filename</i> .
<i>filename</i>	The file whose records you want to list. You can specify <i>filename</i> anywhere in the sentence. LIST.LABEL uses the first word in the sentence that has a file descriptor in the VOC file as the filename.
<i>records</i>	Specifies the records to include in the list. You can specify as many record IDs as you want, separated by spaces. Enclose record IDs in single quotation marks to ensure that Retrieve does not interpret them as field names or keywords.
FROM <i>n</i>	Specifies the records whose record IDs are stored in select list <i>n</i> .

LIST.LABEL Parameters

Parameter	Description								
<i>selection</i>	A selection expression specifying the conditions that data in a record must meet for the record to be selected. A selection expression begins with the keyword WITH. For syntax details, see the WITH keyword in Chapter 2, “UniVerse Keywords.”								
<i>output.limiter</i>	An expression specifying the conditions that values in multi-valued fields must meet for those values to be output. An output-limiting expression begins with the keyword WHEN. For syntax details, see the WHEN keyword in Chapter 2, “UniVerse Keywords.”								
<i>sort</i>	A sort expression that specifies the type of sort and the field to sort on. The syntax is as follows: <table> <tr> <td>BY <i>field</i></td><td>Sort on <i>field</i> in ascending order.</td></tr> <tr> <td>BY.DSND <i>field</i></td><td>Sort on <i>field</i> in descending order.</td></tr> <tr> <td>BY.EXP <i>field</i></td><td>Explode multivalues in <i>field</i> and <i>sort</i> in ascending order.</td></tr> <tr> <td>BY.EXP.DSND <i>field</i></td><td>Explode multivalues in <i>field</i> and <i>sort</i> in descending order.</td></tr> </table> For more information about sort expressions, see Chapter 2, “UniVerse Keywords.” When NLS locales are enabled, LIST.LABEL uses the Collate convention of the current locale to determine the collating order for any existing sort expression. If an index has its own Collate convention defined, LIST.LABEL uses it instead of the current locale definition. For more information, see the <i>UniVerse NLS Guide</i>.	BY <i>field</i>	Sort on <i>field</i> in ascending order.	BY.DSND <i>field</i>	Sort on <i>field</i> in descending order.	BY.EXP <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in ascending order.	BY.EXP.DSND <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in descending order.
BY <i>field</i>	Sort on <i>field</i> in ascending order.								
BY.DSND <i>field</i>	Sort on <i>field</i> in descending order.								
BY.EXP <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in ascending order.								
BY.EXP.DSND <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in descending order.								
<i>output</i>	Specifies the fields whose data you want to list. The fields must be defined in the file dictionary. If you are listing dictionary records, the fields must be defined in the DICT.DICT file. If you do not specify <i>output</i>, fields specified in the @ phrase are listed. If there is no @ phrase in the file dictionary, only record IDs are listed.								

LIST.LABEL Parameters (Continued)

Parameter	Description																
<i>report.qualifiers</i>	One or more of the following keywords: <table><tr><td>COL.HDR.SUPP</td><td>FOOTING</td><td>ID.SUP</td><td>SAMPLE</td></tr><tr><td>COUNT.SUP</td><td>HDR.SUP</td><td>LPTR</td><td>SAMPLED</td></tr><tr><td>DBL.SPC</td><td>HEADING</td><td>NOPAGE</td><td>SUPP</td></tr><tr><td>FIRST</td><td>ID.ONLY</td><td>ONLY</td><td></td></tr></table> These keywords modify the report format. For information about them and their synonyms, see Chapter 2, “UniVerse Keywords.” If you do not specify any report qualifiers, LIST.LABEL produces a report with: ■ UniVerse default heading and footing ■ Paged output to the screen	COL.HDR.SUPP	FOOTING	ID.SUP	SAMPLE	COUNT.SUP	HDR.SUP	LPTR	SAMPLED	DBL.SPC	HEADING	NOPAGE	SUPP	FIRST	ID.ONLY	ONLY	
COL.HDR.SUPP	FOOTING	ID.SUP	SAMPLE														
COUNT.SUP	HDR.SUP	LPTR	SAMPLED														
DBL.SPC	HEADING	NOPAGE	SUPP														
FIRST	ID.ONLY	ONLY															
LIST.LABEL Parameters (Continued)																	

Description

After you enter a LIST.LABEL command, the following prompt appears:

COUNT, ROWS, SKIP, INDENT, SIZE, SPACE [,C] ?

Enter numeric values for each of these specifications (except C) to produce the formatted report. The following list describes what to enter:

Specification	Description
COUNT	The number of labels across the page or screen.
ROWS	The number of lines (fields) displayed or printed per label.
SKIP	The number of blank lines vertically between labels.
INDENT	The number of indented spaces from left margin to the first column of labels. Can be zero.

LIST.LABEL Specifications

Specification	Description
SIZE	The maximum number of characters in any display field.
SPACE	The number of blank spaces horizontally between labels.
C	Do not print empty fields. If you do not specify C, empty fields are printed as a series of blanks.

LIST.LABEL Specifications (Continued)

After you enter these specifications, a series of prompts requests headers for each row in a label. You can enter a header or press ENTER to specify no header.

Row 1 header?**NAME**

In the report these headers appear for each set of labels in the left margin. If INDENT is zero, the command does not prompt for row headers. Be sure to specify an indent area wide enough for the headers.

The total width specifications cannot exceed the capability of the output device, whether it is the terminal screen or the printer. Use the following formula to calculate the total width:

$$\text{INDENT} + \text{COUNT}(\text{SIZE} + \text{SPACE})$$

To produce a continuous report without page breaks, use the [COL.HDR.SUPP](#) keyword. This keyword also suppresses the header at the top of the report.

Example

This example creates labels made up of the record ID, the first name, and the last name. The record IDs are listed by default. The second line is the format specification. COUNT specifies three labels across the page, with each label made up of three fields (ROWS). Two blank lines vertically separate each row of labels (SKIP), the first column of labels is indented 10 spaces from the left (INDENT), each display field is 15 characters wide (SIZE), and 2 blank spaces horizontally separate each column of labels (SPACE).

Because INDENT specifies 10 spaces, LIST.LABEL prompts the user to enter a row header for each row:

```
>LIST.LABEL SUN.MEMBER FNAME LNAME
COUNT, ROWS, SKIP, INDENT, SIZE, SPACE [ ,C ] ? 3,3,2,10,15,2
Row 1 header ID
Row 2 header NAME
Row 3 header <Return>
```

This is the output:

```
LIST.LABEL SUN.MEMBER FNAME LNAME 11:10:31am 20 Oct 1995 PAGE
1
```

ID	2342	3452	4102
NAME	RALPH ADDAMS	JANE SAMUEL	LESLIE BROWN

ID	4108	5390	6100
NAME	HILLARY HENDERSON	ALICE MIX	BOB MASTERS

ID	7100	4309	4439
NAME	ALICE WILLIAMS	EDGAR WILLIAMS	DON ALISON

ID	5205	6203	6783
NAME	ALICE CRATCHETT	SAM YORK	DAVID HALE

ID	7505
NAME	HARRY EDWARDS

13 records listed.

LIST.LOCALES

Use LIST.LOCALES in NLS mode to list information about available NLS locales.

Syntax

LIST.LOCALES { ALL | *locale* [*locale*]... } [DETAIL]

Parameters

The following table describes each parameter of the syntax.,

Parameter	Description
ALL	Lists all defined locales.
<i>locale</i>	Specifies a particular locale.
DETAIL	Lists the name and description of each locale followed by the locale definitions for each category.

LIST.LOCALES Parameters

Description

LIST.LOCALES can use an active select list. If no select list is active, LIST.LOCALES lists all installed locales.

Examples

This example lists the currently defined locales:

```
>LIST.LOCALES

AR-SPANISH      Territory=Argentina, Language=Spanish
CH-FRENCH       Territory=Switzerland, Language=French
CN-CHINESE      Territory=China (PRC), Language=Chinese
DE-GERMAN       Territory=Germany, Language=German
FR-FRENCH       Territory=France, Language=French
US-ENGLISH      Territory=USA, Language=English
```

The next example lists the name and description of the CN-CHINESE locale followed by locale definitions for each category:

>LIST.LOCALES CN-CHINESE DETAIL

```
CN-CHINESE      Territory=China (PRC), Language=Chinese
Time/Date CN-CHINESE
Numeric WIDE.DIGITS
Monetary CHINA
Ctype IDEOGRAPHICS
Collate CHINESE.GB
```

LIST.LOCKS

Use LIST.LOCKS to display the status of the 64 task synchronization locks.

Syntax

LIST.LOCKS

Description

LIST.LOCKS displays the lock number and a number that indicates which user set the lock. If a lock is not set, LIST.LOCKS displays a series of dashes.

Examples

This example first shows the LIST.LOCKS report with no locks set. Next, the user sets lock 5 with the [LOCK](#) command. The second LIST.LOCKS report shows that user 34 set lock 5.

```
1  >LIST.LOCKS
0:----- 1:----- 2:----- 3:----- 4:----- 5:----- 6:-----
7:-----
8:----- 9:----- 10:----- 11:----- 12:----- 13:----- 14:-----
15:-----
16:----- 17:----- 18:----- 19:----- 20:----- 21:----- 22:-----
23:-----
24:----- 25:----- 26:----- 27:----- 28:----- 29:----- 30:-----
31:-----
32:----- 33:----- 34:----- 35:----- 36:----- 37:----- 38:-----
39:-----
40:----- 41:----- 42:----- 43:----- 44:----- 45:----- 46:-----
47:-----
48:----- 49:----- 50:----- 51:----- 52:----- 53:----- 54:-----
55:-----
56:----- 57:----- 58:----- 59:----- 60:----- 61:----- 62:-----
63:-----

1  >LOCK 5
>LIST.LOCKS
0:----- 1:----- 2:----- 3:----- 4:----- 5:34      6:-----
7:-----
8:----- 9:----- 10:----- 11:----- 12:----- 13:----- 14:-----
15:-----
16:----- 17:----- 18:----- 19:----- 20:----- 21:----- 22:-----
23:-----
```

24:----- 25:----- 26:----- 27:----- 28:----- 29:----- 30:-----
31:-----
32:----- 33:----- 34:----- 35:----- 36:----- 37:----- 38:-----
39:-----
40:----- 41:----- 42:----- 43:----- 44:----- 45:----- 46:-----
47:-----
48:----- 49:----- 50:----- 51:----- 52:----- 53:----- 54:-----
55:-----
56:----- 57:----- 58:----- 59:----- 60:----- 61:----- 62:-----
63:-----

LIST.MAPS

Use LIST.MAPS in NLS mode to list information about available NLS maps.

Syntax

LIST.MAPS { ALL | *mapname* [*mapname*] ... } [DETAIL | BASE]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
ALL	Lists all defined maps.
<i>mapname</i>	Specifies a particular map.
DETAIL	Lists the map definitions in the NLS.MAPS.DESCS file.
BASE	Recursively lists any underlying base maps.

LIST.MAPS Parameters

Description

Maps must be built and installed in shared memory. The listing includes the name and description of each map.

Examples

This example lists the name and description of each installed map:

```
>LIST.MAPS
```

```
ASCII          #Standard ASCII 7-bit set
BIG5           #TAIWAN: "Big 5" standard
GB2312         #CHINESE: EUC as described by GB 2312
ISO8859-1      #Standard ISO8859 part 1: Latin-1
ISO8859-1+MARKS #Standard ISO8859 part 1: Latin-1 for type
               1&19 files with marks
MNEMONICS      #ASCII mnemonics for many Unicodes,
               based on UTF8
MNEMONICS-1    #As for MNEMONICS, but ISO8859-1 capable
```

The next example lists map definitions from the NLS.MAPS.DESCS file for the GB2312 maps and ASCII+C1:

```
>LIST.MAPS GB2312 ASCII+C1 DETAIL
```

```
GB2312         #CHINESE: EUC as described by GB 2312
  Base Map     ASCII+C1
  Map Type     DBCS
  Table ID     GB2312-80
  Display Length 2
ASCII+C1       ASCII 7-bit + C1 control chars
  Base Map     C1-CONTROLS
  Map Type     SBCS
  Table ID     ASCII
  Display Length
  Unknown Seq
  Compose Seq  0
```

The next example lists the names of the maps on which the GB2312 map is based:

```
>LIST.MAPS GB2312 BASE
```

```
GB2312         #CHINESE: EUC as described by GB 2312
ASCII+C1       ASCII 7-bit + C1 control chars
C1-CONTROLS    Standard 8-bit ISO control set, 80-9F
C0-CONTROLS    Standard ISO2022 C0 control set, chars 00-1F+7F
```

LIST.READU

Use LIST.READU to list active file and record locks. These locks are set by certain BASIC statements, UniVerse commands, and SQL statements.

Syntax

LIST.READU [*USER terminal#*] [*EVERY*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
USER	Lists the active locks owned by the user whose terminal number is <i>terminal#</i> .
EVERY	Lists the active group locks in addition to the active file and record locks.

LIST.READU Parameters

Description

The LIST.READU command produces a report containing the following information:

Column Heading	Description
Device	A number that identifies the logical partition of the disk where the file system is located.
Inode	A number that identifies the file that is being accessed.
Netnode	A number that identifies the host from which the lock originated. 0 indicates a lock on the local machine. UNIX. The network node number is the last part of the TCP/IP host number specified in the <i>/etc/hosts</i> file. Windows Platforms. This is either the last part of the TCP/IP host number or the LAN Manager node name, depending on the network transport used by the connection.
Userno	The terminal number identifying the user or phantom process that set the lock.
Pid	A number that identifies the controlling process. The number is not returned for remote files accessed across UV/Net.
Login ID	The login name of the user who set the lock. The name is not returned for remote files accessed across UV/Net.
Item-ID	The record ID of the locked record.
LIST.READU Information	

When the report describes file locks, it contains the following additional information:

Column Heading	Description
Lmode	The number (from 1 through the total number of file lock semaphores set by the parameter FSEMNUM) assigned to the lock, and a code that describes the file lock. The code is one of the following:
FS	Shared file lock.
IX	Shared file lock with intent to acquire an exclusive file lock.
FX	Exclusive file lock.
XU	Exclusive file lock set by CLEAR.FILE
CR	Shared file lock set by RESIZE during concurrent resizing of file.
XR	Exclusive file lock set by RESIZE or UVFIXFILE during file restructuring.

Additional Information for File Locks

When the report describes group locks, it contains the following additional information:

Column Heading	Description
Lmode	The number (from 1 through the number of group lock semaphores set by the parameter GSEMNUM) assigned to the lock, and a code that describes a type of group lock. The code is one of the following:
EX	Exclusive lock used when SQL scans a file for the INSERT, UPDATE or DELETE statement.
SH	Shared lock used when SQL scans a file for a SELECT statement.
RD	Read lock.
WR	Write lock.
IN	System information lock (identifies a record that is locked but has not been written to or read).

Additional Information for Group Locks

Column Heading	Description
G-Address	Logical disk address of group, or its offset in bytes from the start of the file. This address for a type 1 or type 19 file is 1. The G-Address appears in hexadecimal notation in the report.
Record Locks	The number of locked records in the group.
Group RD	Number of readers in the group.
Group SH	Number of shared group locks.
Group EX	Number of exclusive group locks.

Additional Information for Group Locks (Continued)

A list of file and group semaphores might also appear in the report. Semaphores are used for internal concurrency control and indicate that a process is actively changing the file or group lock table.

You can determine the number of semaphores on the system by looking at the *uvconfig* file. The FSEMNUM is the total number of file lock semaphores, and the GSEMNUM is the number of group lock semaphores.

When the report describes record locks, it contains the following additional information:

Column Heading	Description
Lmode	The number (from 1 through the number of group lock semaphores set by the parameter GSEMNUM) assigned to the lock, and a code that describes a type of record lock. The code is one of the following:
RL	Shared record lock
RU	Update record lock

Additional Information for Record Locks

Example

This example is from a UNIX system. On Windows systems, the Login Id field is not included.

>LIST.READU EVERY

Active Group Locks:						Record Group Group			
Group									
Device....	Inode....	Netnode	Userno	Lmode	G-Address.	Locks	...RD	...SH	
...EX									
1638492	342346	0	122	8 IN	389000	1	0	0	
0									
1638534	1060608	0	108	11 IN	1	1	0	0	
0									
Active Record Locks:									
Device....	Inode....	Netnode	Userno	Lmode	Pid	Login	Id	Item-	
ID.....									
1638492	342346	0	122	8 RU	15006	janet	15312		
1638534	1060608	0	108	11 RU	13099	davel	DKL1		

LIST.SICA

Use LIST.SICA to list the contents of the security and integrity constraints area (SICA) of an SQL table.

Syntax

LIST.SICA *tablename* [LPTR] [NO.PAGE]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>tablename</i>	The name of an SQL table.
LPTR	Sends output to the printer.
NO.PAGE	Suppresses automatic paging.

LIST.SICA Parameters

Description

LIST.SICA lists the currently active column, constraint, and permission definitions for the specified table.

You cannot use LIST.SICA to reference a remote file over UV/Net.

Example

This example lists the SICA for TBL4003, which is defined as an SQL table. The example shows column definitions and trigger information. The example assumes trigger programs TRIG1 and TRIG2 have been globally cataloged before executing the CREATE TRIGGER statements.

```
>CREATE TABLE TBL4003 (C1 INT NOT NULL PRIMARY KEY, C2 INT);
Creating Table "TBL4003"
Adding Column "C1"
Adding Column "C2"
>CREATE TRIGGER TRIG1 BEFORE DELETE ON TBL4003 FOR EACH ROW
CALLING '*TRIG1';
Adding trigger "TRIG1"
>CREATE TRIGGER TRIG2 AFTER DELETE ON TBL4003 FOR EACH ROW CALLING
'*TRIG1';
Adding trigger "TRIG2"
>CREATE TRIGGER TRIG3 AFTER INSERT ON TBL4003 FOR EACH ROW CALLING
'*TRIG1';
Adding trigger "TRIG3"
>CREATE TRIGGER TRIG4 BEFORE INSERT OR UPDATE ON TBL4003 FOR EACH
ROW CALLING '*TRIG2';
Adding trigger "TRIG4"
>LIST.SICA TBL4003
```

```
LIST.SICA TBL4003 11:37:24am 21 May 1997 Page 1
=====
Sica Region for Table "TBL4003"
```

```
Schema:          TESTTRG
Revision:        3
Checksum is:     8429
  Should Be:     8429
Size:           388
Creator:         0
Total Col Count: 2
  Key Columns:   1
  Data Columns:  1
Check Count:     0
Permission Count: 0
History Count:   0
```

```
Data for Column "C1"
```

```
Position:        0
Key Position:    1
Multivalued:     No
Not Null:        constraint UVCON_0 Yes
Not Empty:       No
Unique:          No
Row Unique:      No
Primary Key:     Yes
```

Default Type: None
Data Type: INTEGER
Conversion: MD0
Format: 10R
No Default Value Defined
No association defined

Data for Column "C2"

Position: 1
Key Position: 0
Multivalued: No
Not Null: No
Not Empty: No
Unique: No
Row Unique: No
Primary Key: No
Default Type: None
Data Type: INTEGER
Conversion: MD0
Format: 10R
No Default Value Defined
No association defined

Trigger "TRIG4" is enabled, creator is "VMARK\csm".
calls "*TRIG2" for
Row Before Insert Update
Trigger "TRIG3" is enabled, creator is "VMARK\csm".
calls "*TRIG1" for
Row After Insert
Trigger "TRIG2" is enabled, creator is "VMARK\csm".
calls "*TRIG1" for
Row After Delete
Trigger "TRIG1" is enabled, creator is "VMARK\csm".
calls "*TRIG1" for
Row Before Delete

>

LIST.UNION

Use LIST.UNION to compare two saved lists and create a third list made up of elements from list one followed by all elements from list two that did not appear in the first list, and which are not redundant.

Syntax

LIST.UNION [*listname1*] [*options*]

Parameters

listname1 is the name of a saved list in the [&SAVEDLISTS&](#) file. If you do not specify *listname1*, LIST.UNION uses the list whose name is an empty string.

options can be one or more of the following:

Parameter	Description
LPTR	Sends output to the printer.
CRT	Sends output to the terminal.
NOPAGE	If output goes to the terminal, suppresses automatic paging.
OVERWRITING	Overwrites an existing list in the &SAVEDLISTS& file.
NO.WARN	If output goes to the terminal or printer, suppresses line numbers.
HEX	Lists output in hexadecimal format.

LIST.UNION Parameters

Description

The system prompts you to enter the name of a second list by displaying the following prompt:

WITH:

Enter the name of a second list. _

If CRT or LPTR is not specified, the system prompts you for a destination for the result of the LIST.UNION operation, by displaying the following prompt:

TO:

Enter the name of the destination file using one of the following formats:

[*dest.listname* [*account.name*]]
(*dest.file*) *record.ID*

If you use the first form, LIST.UNION copies the original list to *dest.listname*. If you do not specify *dest.listname*, LIST.UNION uses an empty string as the list name. If you specify *account.name*, LIST.UNION creates the new list in the [&SAVEDLISTS&](#) file of the specified account. If you specify OVERWRITING, the new list overwrites any existing list with the same name.

If you use the second form, LIST.UNION converts the list to a record in *dest.file*, and each element in the list constitutes a field in the record. If the size of the list exceeds 32,267 bytes, LIST.UNION truncates the record.

NLS Mode

When locales are enabled, LIST.UNION uses the Collate convention of the current locale to determine the collating order for the new list. For more information, see the *UniVerse NLS Guide*.

Example

This example creates and saves two select lists: 1991 and 1992. Comparing the two saved lists, LIST.UNION creates and saves a third select list, UNION.92, which contains all elements from list 1991 followed by those elements in list 1992 that are not in list 1991 and that are not redundant.

```
>SELECT SUN.MEMBER WITH YR.JOIN EQ "1991"  
2 record(s) selected to SELECT list #0.  
>>SAVE.LIST 1991  
2 record(s) SAvEd to SELECT list "1991".  
>SELECT SUN.MEMBER WITH YR.JOIN >= "1991"  
7 record(s) selected to SELECT list #0.  
>>SAVE.LIST 1992  
7 record(s) SAvEd to SELECT list "1992".  
>LIST.UNION 1991  
WITH: 1992  
TO: UNION.92  
7 record(s) SAvEd to SELECT list "UNION.92".
```

LISTME

Use LISTME to list your currently running processes.

Syntax

LISTME

Description

The contents of the LISTME report depends on the operating system you are running.

UNIX

LISTME lists the following information:

Column Heading	Description
USER or UID	Your UNIX login name
PID	Process ID number
PPID	Process ID number of the parent process
C	Scheduler status
STIME	The time the process was started
TTY	Terminal number
TIME	Accumulated CPU time
CMD	The command that the process is running

LISTME Information on UNIX

LISTME and *ps* produce this report in different formats depending on the UNIX system on which you are working.

Windows Platforms

LISTME lists the following information:

Column Heading	Description
UID	Your user identifier. If you log in as a member of the Administrators' group, this value is 0. If you log in as the "Guest" user, this value is -2. Any other users are assigned a unique UID number.
User No	UniVerse user number.
User Name	Domain and user name.
Terminal No	User type (for example, console, telnet, and so on) and UniVerse user number.
Login Time	Date and time the process was started.

LISTME Information on Windows Platforms

Examples

UNIX. The LISTME command executes the UNIX *df* command, which produces a report similar to this:

```
>LISTME
USER  PID  PPID  C    STIME  TTY    TIME CMD
sue  20343  15881  12   11:52:12 ttyq9   0:02 STATUS ME
sue  20344  20343  47   11:52:14 ttyq9   0:01 ps -f
sue  15881  15877  0    09:26:42 ttyq9   0:31 uv
sue  15877  15876  0    09:26:33 ttyq9   0:03 -csh
>
```

Windows Platforms. The LISTME command produces a report similar to this:

```
>LISTME
These are the UniVerse processes running under your user id:
UID User Name User  Terminal No      Login Time
0   136 MK33 mith  console:136      15/01/96 09:33:10
0   143 MK33 mith  telnet:143       15/01/96 11:14:49
>
```

LISTU

Use LISTU to list all the users currently logged in.

Syntax

LISTU

Description

The contents of the LISTU report depends on the operating system you are running.

Examples

UNIX

The LISTU command executes the UNIX *who* command, which produces a report similar to this:

```
1  >LISTU
These are the users presently sharing the system.

larry      tty0      Feb 17 16:54
rd         tty1      Feb 18 07:53
ellen      tty2      Feb 18 09:55
glen       tty3      Feb 18 08:03
tom        tty4      Feb 18 11:29
ted        tty5      Feb 18 08:11
qa         tty6      Feb 18 08:32
qa         tty7      Feb 17 17:19
wendy      tty8      Feb 18 08:37
steve      tty9      Feb 18 08:41
donna      ttya      Feb 18 08:42
sean       ttyb      Feb 18 08:47
walter     ttyc      Feb 18 12:28
katy       ttyd      Feb 18 08:50
adrian     ttye      Feb 18 08:52
qa         ttyf      Feb 18 09:04
tina       ttyq0     Feb 18 08:56
```

```
bill          ttyq1          Feb 18 08:56
.
.
.
There are currently 55 users logged on the system.
>
```

Windows Platforms

The LISTU command produces a report similar to this:

```
>LISTU
These are the UniVerse users presently sharing the system:
```

UID	User Name	User	Terminal No	Login Time
0	136	MK33\Smith	console:136	15/01/96 09:33:10
0	143	MK33\Smith	telnet:143	15/01/96 11:14:49
1720	185	MC04\Whyte	telnet:185	15/01/96 11:23:43
-2	192	MK33\Scott	console:192	15/01/96 15:44:38
1984	206	VMARK\Adams	console:206	15/01/96 14:12:08

```
There are currently 5 users logged on the system.
>
```

LIST

Use the LIST commands to view the contents of the VOC file. The LIST commands all begin with LIST followed by one or more letters indicating the VOC entries you want to list. For example, LISTF lists files and LISTK lists keywords.

Syntax

LISTDOS

LISTF

LISTFL

LISTFR

LISTK

LISTM

LISTO

LISTPA

LISTPH

LISTPQ

LISTR

LISTS

LISTSL

LISTUNLISTPH

LISTV

Description

The LIST commands are sentences or procs stored in the UNIVERSE.VOCLIB file. The following table describes each command:

Command	Description
LISTDOS	Lists all DOS commands in the VOC file.
LISTF	Lists all files defined in the VOC file.
LISTFL	Lists files stored in this account.
LISTFR	Lists files stored in remote accounts.
LISTK	Lists all keywords in the VOC file.
LISTM	Lists all menu selector records in the VOC file.
LISTO	Lists “other”-type keywords in the VOC file.
LISTPA	Lists all paragraphs stored in the VOC file.
LISTPH	Lists all phrases stored in the VOC file.
LISTPQ	Lists all proc commands in the VOC file.
LISTR	Lists all remote commands in the VOC file.
LISTS	Lists all stored sentences in the VOC file.
LISTSL	Lists all verbs in the VOC file that use a select list.
LISTUN	Lists all UNIX commands in the VOC file.
LISTV	Lists all verbs in the VOC file.

LIST Commands

LOCK

Use LOCK to set any of the 64 task synchronization locks to the locked state.

Syntax

LOCK *n* [PROMPT | NO.WAIT]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>n</i>	Any number from 0 through 63.
PROMPT	If the lock is not available, returns control to the command processor so that you can execute other commands. When the lock is available, LOCK locks it and sends a message indicating that the lock is set. If NOTIFY ON is set, the message appears immediately, otherwise it appears when the next UniVerse prompt (>) appears (see NOTIFY for more information).
NO.WAIT	If the lock is not available, returns control to the command processor and issues a message. You must use the LOCK command again later to set the lock.

LOCK Parameters

Description

If you use LOCK without the PROMPT or NO.WAIT keywords, and if the lock you specify is not available, the command waits until the lock is free before returning control to the command processor.

If the lock you specify is already locked by someone else, LOCK acts differently depending on which qualifier you specify.

LOCK performs the same function as the BASIC LOCK statement.

Use the [CLEAR.LOCKS](#) command to clear task synchronization locks. Use the [LIST.LOCKS](#) command to display the lock table.

Examples

This example first shows the LIST.LOCKS report with no locks set. Next, the user sets lock 5 with the LOCK command. The second LIST.LOCKS report shows that UniVerse user 34 set lock 5.

```
>LIST.LOCKS
 0:----- 1:----- 2:----- 3:----- 4:----- 5:----- 6:-----
 7:-----
 8:----- 9:----- 10:----- 11:----- 12:----- 13:----- 14:-----
15:-----
16:----- 17:----- 18:----- 19:----- 20:----- 21:----- 22:-----
23:-----
24:----- 25:----- 26:----- 27:----- 28:----- 29:----- 30:-----
31:-----
32:----- 33:----- 34:----- 35:----- 36:----- 37:----- 38:-----
39:-----
40:----- 41:----- 42:----- 43:----- 44:----- 45:----- 46:-----
47:-----
48:----- 49:----- 50:----- 51:----- 52:----- 53:----- 54:-----
55:-----
56:----- 57:----- 58:----- 59:----- 60:----- 61:----- 62:-----
63:-----

>LOCK 5
>LIST.LOCKS
 0:----- 1:----- 2:----- 3:----- 4:----- 5:34      6:-----
 7:-----
 8:----- 9:----- 10:----- 11:----- 12:----- 13:----- 14:-----
15:-----
16:----- 17:----- 18:----- 19:----- 20:----- 21:----- 22:-----
23:-----
24:----- 25:----- 26:----- 27:----- 28:----- 29:----- 30:-----
31:-----
32:----- 33:----- 34:----- 35:----- 36:----- 37:----- 38:-----
39:-----
40:----- 41:----- 42:----- 43:----- 44:----- 45:----- 46:-----
47:-----
48:----- 49:----- 50:----- 51:----- 52:----- 53:----- 54:-----
55:-----
56:----- 57:----- 58:----- 59:----- 60:----- 61:----- 62:-----
63:-----
```

LOG.RESTORE

Use LOG.RESTORE to restore transaction logging log files from tape to disk. You must be a UniVerse Administrator logged in to the UV account to use this command.

Syntax

LOG.RESTORE *first.log last.log* [*logdir.path*] [*device*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>first.log</i>	The number of the first log file to restore.
<i>last.log</i>	The number of the last log file to restore.
<i>logdir.path</i>	The full path of the directory into which log files are to be restored. If you do not specify a directory path, the current log directory is used.
<i>device</i>	The device name of the tape drive. If you do not specify <i>device</i> , the device defined as MT0 in the &DEVICE& file is used.

LOG.RESTORE Parameters

Description

Use LOG.RESTORE to restore log files that you have backed up to tape and removed from the log directory. LOG.RESTORE does not change anything in the UV_LOGS file.

Example

This example restores four log files—*lg344*, *lg345*, *lg346*, *lg347*—to the directory */u2/logrestore* from the default tape drive:

>LOG.RESTORE 344 347 /u2/logrestore

LOG.SAVE

Use LOG.SAVE to save transaction logging log files from disk to tape. You must be a UniVerse Administrator logged in to the UV account to use LOG.SAVE.

Syntax

LOG.SAVE *first.log last.log* [*logdir.path*] [*device*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>first.log</i>	The number of the first log file to save.
<i>last.log</i>	The number of the last log file to save.
<i>logdir.path</i>	The full path of the directory where the log files reside. If you do not specify a directory path, the current log directory is used.
<i>device</i>	The device name of the tape drive. If you do not specify <i>device</i> , the device defined as MT0 in the &DEVICE& file is used.

LOG.SAVE Parameters

Description

Use LOG.SAVE to save transaction logging files from disk to tape. LOG.SAVE does not change anything in the UV_LOGS file.

Example

This example saves four log files—*lg344*, *lg345*, *lg346*, *lg347*—from the directory */u2/logrestore* to the default tape drive:

>LOG.SAVE 344 347 /u2/logsave

LOGIN

Use LOGIN to initialize your UniVerse account environment.

Syntax

LOGIN

Description

When your UniVerse account is first created, the VOC file does not contain a LOGIN entry. You can create or change a LOGIN entry in the VOC file at any time. UniVerse searches for this entry and executes it when you invoke UniVerse and when you log to a UniVerse account.

A LOGIN entry must be a sentence, a paragraph, a menu, a proc, a BASIC program, or a verb. Usually the LOGIN entry calls a menu, sets printer parameters (with the SETPTR command), calls a security or system accounting program, or calls a program.

When you log to a UniVerse account by executing a [LOGTO](#) or [CHDIR](#) command or when you start a phantom process, the UniVerse command processor executes the LOGIN entry.

To prevent a phantom from executing the LOGIN entry, add a test to the LOGIN entry for @TTY = "phantom". For example, if a phantom executes a LOGIN paragraph containing the following IF statement, the IF statement exits the LOGIN paragraph:

```
IF @TTY = 'phantom' THEN GO END.OF.LOGIN
```

Put commands that phantoms should not execute in the last section of the LOGIN paragraph. The last line of the LOGIN paragraph must be the following label:

```
END.OF.LOGIN:
```

For more information about phantom processes, see the [PHANTOM](#) command.

If you want a UniVerse process with a single command parameter, such as `uv "UPDATE .MASTER"`, to avoid executing the LOGIN entry, use the `@TTY` test just described, and have the process direct standard input, standard output, and standard error to nonterminal devices. For example:

```
uv "UPDATE.MASTER" </dev/null >/dev/null 2>&1
```

The preceding example is executed from a Bourne shell.

Example

This paragraph automatically starts up the ACCOUNTING program when you log in to UniVerse:

```
      LOGIN
0001: PA
0002: RUN BP ACCOUNTING
```



LOGON

Use LOGON to begin a UniVerse session on another terminal or to run a UniVerse command on another terminal.

***Note:** This command is not supported on Windows platforms.*

Syntax

LOGON *port* [*command*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>port</i>	The name of the terminal on which to start up UniVerse. Prefix <i>port</i> with the string TTY (that is, TTY <i>port</i>) to make up the name of the port definition record, which must exist in the &DEVICE& file.
<i>command</i>	The command text. The command can be a stored sentence or paragraph or a UniVerse command. It cannot exceed 25 characters.

LOGON Parameters

Description

If you enter **LOGON** *port*, a UniVerse session starts on the named terminal.

If you enter **LOGON** *port command*, a UniVerse job starts that runs only the specified command. Standard input, standard output, and standard error are redirected to the device name in [&DEVICE&](#), and a phantom process starts.

You can control jobs started with LOGON using the usual methods of controlling phantom processes (see your UNIX documentation). For information about phantom processes, see the [PHANTOM](#) command.

LOGOUT

Use LOGOUT or LO to log out of the UniVerse environment or to kill a phantom process.

Syntax

LO [*-pid#*]

LOGOUT [*-pid#*]

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>-pid#</i>	The process number of a phantom process to log out.

LOGOUT Parameter

Description

UNIX System V

LOGOUT returns you to the system login prompt. When you use LOGOUT as a verb, LOGOUT is a synonym for the OFF command.

UNIX Berkeley

LOGOUT works like QUIT.

Windows Platforms

The way this command works depends on how you accessed UniVerse. If you accessed UniVerse from a Windows command window, LOGOUT terminates the UniVerse session and returns control to the command window. If you accessed UniVerse from a telnet interface, LOGOUT terminates UniVerse and the telnet session.

When you log out, UniVerse closes all files that you have been using, releases any devices assigned to your terminal, and discards the sentence stack if the VOC file STACKWRITE entry is OFF.

The login name for the *-pid#* qualifier must be the same as your login name. For more information about phantom processes, see the [PHANTOM](#) command.

LOGTO

Use LOGTO to log out of one UniVerse account and log in to another UniVerse account without leaving UniVerse. The current process continues in the new account.

Syntax

LOGTO *account*

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>account</i>	The name of the account to which you want to log in. You can specify the name of the account as it is defined in the UV.ACCOUNT file, as a login name, or as a path.

LOGTO Parameter

Description

If UniVerse does not recognize the account name as an existing UniVerse account, it displays the following message:

"account" is not a valid account name.

If the directory is not set up for UniVerse, it displays the following message:

Account *"directory"* is not set up for UniVerse.

LOGTO retains the current sentence stack.

If the account you are logging to has a LOGIN entry in its VOC file, UniVerse executes it before displaying the > prompt.

LOGTO.ABORT

Use LOGTO.ABORT to log out of one UniVerse account and log in to another UniVerse account. LOGTO.ABORT aborts the process from which it is called.

Syntax

LOGTO.ABORT *account*

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>account</i>	The name of the account to which you want to log in. You can specify the name of the account as it is defined in the UV.ACCOUNT file, as a login name, or as a path.

LOGTO.ABORT Parameter

Description

If UniVerse does not recognize the account name as an existing UniVerse account, it displays the following message:

"account" is not a valid account name.

If the directory is not set up for UniVerse, it displays the following message:

Account *"directory"* is not set up for UniVerse.

LOGTO.ABORT retains the current sentence stack.

If the account you are logging to has a LOGIN entry in its VOC file, UniVerse executes it before displaying the > prompt.

LOOP

Use LOOP in a paragraph to define a range of sentences to be executed repeatedly. LOOP begins the range, and REPEAT ends the range.

Syntax

LOOP

statements

REPEAT

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>statements</i>	UniVerse statements you want to execute repeatedly.

LOOP Parameter

Description

You can use any number of LOOP...REPEAT loops. At least one of the statements must be an IF statement that specifies the exit condition and action.

You can use LOOP...REPEAT loops only in paragraphs. They have no function on the command line.

Example

This example represents a paragraph as it appears in a VOC file:

```
001 PA
002 LOOP
003 IF <<A, ENTER VENDOR>> = '' THEN GO END
004 LIST PAYABLES <<A, ENTER VENDOR>> AMT.PAID PYMT.DATE
005 REPEAT
006 END: DISPLAY Done
```

The paragraph runs the LIST statement (line 4) repeatedly until the user enters nothing at the ENTER VENDOR prompt (by pressing ENTER). The IF statement (line 3) then transfers control to line 6, which displays the message Done and returns the user to the system prompt (>).



MAIL

Use MAIL to communicate with other users on the system.

***Note:** This command is not supported on Windows platforms.*

Syntax

MAIL [*-s subject*] [*user.list*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>subject</i>	The subject of your message. MAIL uses only the first word after the <i>-s</i> flag as the subject. Enclose subjects containing spaces in quotation marks.
<i>user.list</i>	A list of the user names of the people you want to send the message to. Enter user names exactly as users do when they log in.

MAIL Parameters

Description

You can send mail to other people on the system, and you can also receive mail that other people send you.

To send mail, enter **MAIL** and the user name of the person to whom you want to send mail, then type the text of your message. To transmit the message, press Ctrl-D at the beginning of a new line. If you change your mind before you send the message, press your Intr (interrupt) key twice to cancel the message.

To see if you have any mail messages, enter **MAIL** with no arguments. The system tells you if you have unread mail. If you do not have mail, you return to the UniVerse prompt. If you have mail, MAIL prints a list of message headers telling who sent each message and when, followed by the UNIX mail prompt (&). For a list of mail commands, enter ? at this prompt.

Capitalization is significant for mail commands. Most mail commands must be typed as lowercase letters. Some of the basic mail commands are as follows:

Command	Description
[t] [<i>message list</i>]	Displays the specified messages.
n	Goes to next message and displays it.
e [<i>message list</i>]	Edits messages.
f [<i>message list</i>]	Lists heading lines of specified messages.
d [<i>message list</i>]	Deletes specified messages.
s [<i>message list</i>] <i>file</i>	Appends the specified messages to <i>file</i> .
u [<i>message list</i>]	Undeletes specified messages.
r [<i>message list</i>]	Replies to specified messages.
pre [<i>message list</i>]	Makes specified messages go back to <i>/usr/mail</i> .
m [<i>user list</i>]	Sends mail to specific users.
q	Quits, saving unresolved messages in <i>mbox</i> .
x	Quits, does not remove system mailbox.
h	Lists active message headers.
?	Lists a brief summary of commands.
!	Shell escape.
c [<i>directory</i>]	Changes directory.

MAIL Commands

message list is a list of message numbers or user names separated by spaces. If no message list is specified, the command applies to the current message.

For more information on additional mail commands and how mail works, see your UNIX documentation.

Examples

In the following example, the user invokes MAIL to send a message to Ken, whose login name is *ken*. The message is two lines long. The Ctrl-D at the beginning of the third line ends the message and sends it to Ken.

```
>MAIL ken<Return>
HELLO<Return>
HOW ARE YOU TODAY<Return>
<Ctrl-D>
EOT
```

When Ken invokes MAIL, the new message is listed. When he enters **n** (next), MAIL displays the message headers followed by the text of the message. When he enters **q** (quit), the message is saved in the file *mbox*, and Ken returns to the UniVerse prompt.

```
>MAIL
12:13:14 01 JUL 1995
Mail version 2.18 5/19/95.  Type ? for help.
"/usr/spool/mail/ken": 1 message 1 new
>N  1 ken Sat Jul  1 12:13  13/241
&n
Message 1:
From bob Sat Jul  1 12:13:14 1995
Received: by aragorn.UUCP (4.12/4.7)
        id AA01966; Wed, 1 Jul 95 12:13:12 edt
Date: Sat, 1 Jul 95 12:13:12 edt
From: bob (documentation account - Bob Bush)
To: ken
Status: R

HELLO
HOW ARE YOU TODAY

>q
Saved 1 message in mbox
>
```

MAKE.DEMO.FILES

Use MAKE.DEMO.FILES to initialize 10 sample files for the Circus database in the account in which you are logged in.

Syntax

MAKE.DEMO.FILES

Description

UniVerse supplies the sample files ACTS, CONCESSIONS, ENGAGEMENTS, EQUIPMENT, INVENTORY, LIVESTOCK, LOCATIONS, PERSONNEL, RIDES, and VENDORS. MAKE.DEMO.FILES creates the data files and the file dictionaries for the 10 files, compiles the dictionaries, and loads the data (from the *sample* directory in uvhome) into them. The 10 filenames all have the .F suffix. The data in the .F files is the same as the data in the .T tables created by [MAKE.DEMO.TABLES](#).

MAKE.DEMO.FILES verifies that files with these names do not exist. If a file with one of these names exists, MAKE.DEMO.FILES displays a message that the file already exists and MAKE.DEMO.FILES exits without taking any action.

Files are created with truncated names if they exceed the system limit.

MAKE.DEMO.TABLES

Use MAKE.DEMO.TABLES to create and load the 10 sample tables for the Circus database in the account in which you are logged in. The account must be a UniVerse SQL schema, and you must be a UniVerse SQL user defined in the SQL catalog.

Syntax

MAKE.DEMO.TABLES

Description

UniVerse supplies the sample tables ACTS, CONCESSIONS, ENGAGEMENTS, EQUIPMENT, INVENTORY, LIVESTOCK, LOCATIONS, PERSONNEL, RIDES, and VENDORS. MAKE.DEMO.TABLES creates the 10 tables and loads the data into them (from the *sample* directory in uvhome). The 10 table names all have the .T suffix. You are the owner of these tables.

MAKE.DEMO.TABLES verifies that your account is a UniVerse SQL schema. If it is not, MAKE.DEMO.TABLES tells you that a DBA needs to make the account into a schema by using the command [SETUP.DEMO.SCHEMA](#).

MAKE.DEMO.TABLES then verifies that you are a UniVerse SQL user and that tables with these names do not exist.

MAKE.MAP.FILE

Use MAKE.MAP.FILE to create a UniVerse file called **&MAP&**, which contains a map of the system catalog space.

Syntax

MAKE.MAP.FILE

Description

The information in the **&MAP&** file is the same as that displayed by the **MAP** command. By creating the **&MAP&** file, you can take full advantage of the selection, sorting, and report formatting capabilities of the Retrieve commands to produce reports on the catalog space.

The following information is stored in the **&MAP&** file:

Field	Description
NAME	Name of the program or subroutine
DATE	The date it was last accessed
WHO	The account name from which it was cataloged
VAR	The number of variables
REF	The number of times the program has been run since being cataloged
OBJ	The number of bytes of object code
SEG	The number of common segments
CR	The size of the cross-reference table in bytes
SIZE	The total number of bytes for object code, cross-reference table, and symbol table
M/S	The type of program cataloged; M for a main program, S for a subroutine

Information Stored in &MAP& File

Field	Description
REV	The BASIC compiler release level number
CONSTANTS	The number of constants in the program
ARGS	The number of arguments if the program is a subroutine
COMMON.VARS	The number of variables in unnamed common
SHM	The program is or is not loaded from shared memory

Information Stored in &MAP& File (Continued)

The @ phrase for the &MAP& file contains the fields NAME, DATE, WHO, VAR, REF, OBJ, SEG, CR, and SIZE. All the fields are contained in the @LPTR phrase.

The &MAP& file is usually defined only in the UV account, with pointers to it from other accounts. You can, however, create your own copy of this file. The dictionary must be the same as the dictionary of the UV account &MAP& file.

The contents of the &MAP& file become outdated as soon as anyone uses a command that changes the catalog, such as [CATALOG](#) or [DELETE.CATALOG](#). We suggest that you use MAKE.MAP.FILE immediately before you produce a report about the catalog. This practice ensures that your information accurately reflects the current catalog.

MAP

Use MAP to display the contents of the system catalog space. MAP displays each program and subroutine in the catalog alphabetically.

Syntax

MAP

Description

MAP displays the following information for each item:

Column Heading	Description
Catalog Name	The name of the cataloged program or subroutine
Date	The date the program was last accessed
Who	The account name from which the program was cataloged
Ref	The number of times the program has been run since being cataloged
Var	The number of variables
Seg	The number of common segments
Obj	The number of bytes of executable object code
CR	The size of the cross-reference table in bytes
Size	The total number of bytes, consisting of the executable object code, the cross-reference table, and the symbol table

MAP Display

See [MAKE.MAP.FILE](#) for an alternative to examining the system catalog space.

MASTER

Use MASTER to release some or all of the 64 task synchronization locks set with the [LOCK](#) command, to enable the Break key for a specified user, or to log users out of UniVerse. You must be aUniVerse Administrator logged in to the UV account to use MASTER.

Syntax

MASTER { LOCKS | BREAK | OFF } *user.no* | ALL

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
LOCKS	Releases task synchronization locks.
BREAK	Enables the Break key.
OFF	Logs out users.
<i>user.no</i>	The user number of the user whose locks are released, whose Break key is enabled, or who is logged out. Use the PORT.STATUS command to get a list of the user numbers.
ALL	Specifies that the MASTER command affects all users.

MASTER Parameters

Description

The MASTER command:

- Does not kill itself or its parent processes
- Releases locks regardless of who set them
- Does not release record locks or file locks

MENU.DOC

Use MENU.DOC to produce a menu listing.

Syntax

MENU.DOC

Description

MENU.DOC prompts you to enter the name of a file that stores menu records, and the name of a menu record.

For each option of the specified menu, MENU.DOC lists the menu's contents. The listing contains the following:

- Number and text of the menu option.
- Explanation field containing additional information about the option.
- Expanded VOC file entry that the menu executes when a user selects the option. If the Action field of a menu is the name of a sentence or paragraph, MENU.DOC displays the sentence or paragraph. If the Action field is the name of a menu pointer, MENU.DOC displays the name of the menu file and the menu record.

MENU.DOC is the same as option 15 on the UniVerse Menu Maintenance menu, which the [MENUS](#) command invokes.

See [MENU.PIX](#) for information about how to print a menu on the printer.

MENU.PIX

Use MENU.PIX to print an image of a menu on the printer.

Syntax

MENU.PIX

Description

MENU.PIX prompts you as follows:

Name of the Menu file =

Enter the name of a file that stores menu records. MENU.PIX then prompts you with the following:

Name of the Menu =

Enter the name of the menu record you want to print. MENU.PIX prints the menu and prompts you to enter the name of another menu record. If you do not want to print another menu record, press ENTER. MENU.PIX prompts you to enter the name of another menu file. If you do not want to print any more menu records, press ENTER. You return to the UniVerse prompt.

Using [MENU.DOC](#) and [MENU.PIX](#), you can quickly and easily document menus you create for your system. MENU.DOC lists menu option details, and MENU.PIX prints a copy of the menu record.

MENUS

Use MENUS to create, update, and maintain UniVerse menus.

Syntax

MENUS

Description

The MENUS command invokes the menu MAKE.MENUS in the UNIVERSE.MENU.FILE. The menu displays the following options:

1. Enter/Modify	a menu
2. Enter/Modify	a formatted menu
3. Display	a summary of all menus on a menu file
4. Display	the contents of a menu
5. Enter/Modify	a VOC menu selector item
6. Enter/Modify	a VOC stored sentence item
7. Display	all menu selector items on the VOC file
8. Display	all stored sentence items on the VOC file
9. Display	the dictionary of a file
10. Print	a summary of all menus on a menu file
11. Print	the contents of a menu
12. Print	a virtual image of a menu on the printer
13. Print	the contents of a formatted menu
14. Print	the dictionary of a file
15. Print	detail of a menu, including VOC records referenced

Option 15 is also available as the UniVerse command [MENU.DOC](#).

MERGE.LIST

Use MERGE.LIST to merge two numbered select lists using relational set operations and put the result in a third select list.

Syntax

```
MERGE.LIST list1 { UNION | INTERSECT [ ION ] | DIFF [ ERENCE ] } list2
[ TO list3 ] [ COUNT.SUP ]
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>list1</i> <i>list2</i>	The numbers, 0 through 10, of the select lists you want to merge.
UNION	<i>list3</i> contains all elements from <i>list1</i> and all elements from <i>list2</i> that are not in <i>list1</i> .
INTERSECT [ION]	<i>list3</i> contains all elements from <i>list1</i> that are also in <i>list2</i> .
DIFF [ERENCE]	<i>list3</i> contains all elements from <i>list1</i> that are not in <i>list2</i> .
<i>list3</i>	The select list number for the output select list. If you do not specify <i>list3</i> , select list 0 is used.
COUNT.SUP	Suppresses the display of the number of records selected.

MERGE.LIST Parameters

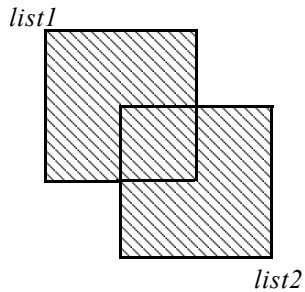
Description

Use MERGE.LIST to create a select list from existing select lists. This avoids the need to execute another [SELECT](#) command.

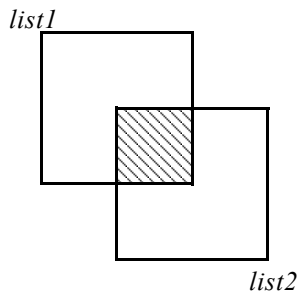
You must specify the numbers of the two select lists to be merged. If you specify a select list that is not active or is empty, MERGE.LIST issues a warning message and terminates.

If MERGE.LIST is successful, the two numbered select lists are emptied, and @SYSTEM.RETURN.CODE returns the number of elements in the resulting select list. If MERGE.LIST is not successful, the select lists are emptied and @SYSTEM.RETURN.CODE is set to -1. If a syntax error occurs, the lists are not changed.

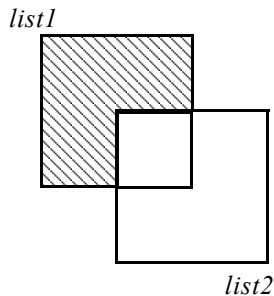
The relational set operations are illustrated in the following example. The shaded areas indicate what is included in the merged list.



UNION produces a select list whose elements are contained in *list1*, *list2*, or both. *list3* contains no duplicates, and the order of select list elements is not defined.



INTERSECTION produces a select list whose elements are contained in *list1* and *list2*. *list3* contains no duplicates, and the order of select list elements is not defined.



DIFFERENCE produces a select list whose elements are the remainder of *list1* after the elements of *list2* that are also in *list1* are subtracted from it. *list3* contains no duplicates, and the order of select list elements is not defined.

Figure 0-1. MERGE.LIST Results

MESSAGE

Use MESSAGE on a UNIX system to send a message to any user currently logged in to the system, or to the system console. Use MESSAGE on a Windows system to send a message to any users logged in to UniVerse. You can also use MESSAGE to enable or disable receive mode for all messages.

Syntax

MESSAGE [*user* | *number* | -ACCEPT | -REJECT | -MSG *text* | STATUS [ME] | -SUPPRESS]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>user</i>	The user name of the person to whom you want to send a message. Windows Platforms. You can specify a full <i>domain\username</i> which sends a message only to the specified user in the specified domain. If you specify just a user name, the message goes to all users with the same user name in all domains. If you specify just a domain, the message goes to all users in that domain. If you do not specify <i>user</i> , MESSAGE sends the message to the system console.
<i>number</i>	UNIX. The terminal number specified in the “tty” column in output from the STATUS USERS command. Windows Platforms. The UniVerse user number specified in the output from a LISTU command.
-ACCEPT	Enables receive mode for all messages.
-REJECT	Disables receive mode for all messages.
-MSG	Specifies the message <i>text</i> .

MESSAGE Parameters

Parameter	Description
-STATUS	Displays the status of receive mode for all users.
ME	Displays the status of receive mode for all users with your user name.
-SUPPRESS	Displays the message on one line, suppressing the “Message from” line and the trailing carriage return and linefeed.

MESSAGE Parameters (Continued)

Description

If you do not use the -MSG option, MESSAGE prompts you to enter the text of your message. Pressing ENTER ends the message and sends it.

You cannot use the -ACCEPT, -REJECT, or -STATUS options if you are using MESSAGE in a phantom process.

Example

```
>MESSAGE
TO OPERATOR(console) PLEASE MOUNT TAPE1<Return>
message to OPERATOR(console) sent

>MESSAGE pts/16
TO lee(pts/16) PLEASE LOG OUT<Return>
message to lee(pts/16) sent
```

MKFILELIST

Use MKFILELIST to create a saved select list of files to activate for transaction logging. You must be a UniVerse Administrator logged in to the UV account to use MKFILELIST.

Syntax

MKFILELIST *account listname*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>account</i>	The name of the account whose files you want to list. Define <i>account</i> in the UV.ACCOUNT file in the UV account.
<i>listname</i>	The record ID of the saved select list.

MKFILELIST Parameters

Description

MKFILELIST creates a select list and saves it in the [&SAVEDLISTS&](#) file. Each line in the saved list contains the account name and a filename, separated by a colon. MKFILELIST lists all files defined in the VOC file of the specified account, except for files whose names begin with ampersand (&).

The saved list includes type 1 and type 19 files defined in your account. You may want to remove the names of these files from the list because you cannot activate or deactivate such files for transaction logging.

You can use the [EDIT.LIST](#) command to add or delete files in this list. The files in the list need not all be in the same account.

Use the [ACTLIST](#) command to activate the files in the list for transaction logging.

Example

This example creates a saved select list called INV.FILE.LIST in the &SAVEDLISTS& file. Each file in the INVENTORY account is listed on a separate line in the format INVENTORY: *filename*.

```
>MKFILELIST INVENTORY INV.FILE.LIST
```

The saved list looks like this:

```
INV.FILE.LIST
0001: INVENTORY:APP.PROGS
0002: INVENTORY:APP.PROGS.O
0003: INVENTORY:BASIC.HELP
0004: INVENTORY:BIN
0005: INVENTORY:BLTRS
0006: INVENTORY:BOOKS
0007: INVENTORY:BP
0008: INVENTORY:BP.L
0009: INVENTORY:BP.O
0010: INVENTORY:CLASSES.DOC
0011: INVENTORY:CUSTOMERS
.
.
.
```

MOTIF

Use MOTIF to display a UniVerse menu in Motif format.

Syntax

MOTIF *menu.name*

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>menu.name</i>	The record ID of the VOC menu entry that invokes the menu. The options listed on the menu that you invoke must call submenus.

MOTIF Parameters

Description

When you display a menu in Motif format, the title of the menu appears at the top of the screen. Under the title, options of the main menu are displayed in a menu bar. The first menu option is highlighted. Each option in the menu bar stands for a pull-down menu that scrolls down from the main menu bar when the user selects an option.

Menus list actions that can be performed and options that display submenus (*cascading* menus). Options followed by an arrow (=>) display a submenu.

The options on the submenus are names or descriptions of UniVerse sentences.

To display a submenu, highlight the option, then select it. To start an action, simply select it. If a data entry screen appears, instructions in the lower part of the screen prompt you to enter the appropriate data. System messages also appear at the lower part of the screen.

Certain responses work at all levels of a Motif menu system. At any menu bar, use the Right Arrow key (→) and the Left Arrow key (←) to move the highlight to the option you want.

Note that → does one of two things. If the highlighted menu item calls a submenu, pressing → displays the submenu. If the item does not call a submenu, pressing → displays the submenu belonging to the next option on the main menu bar.

To select the highlighted option, press ENTER or Spacebar. To move to an option and select it with one keystroke, press the capitalized letter (*mnemonic*) of the option you want. The mnemonic is not always the first letter of an option. If your keyboard does not have arrow keys, you can use the mnemonic letters to move around the menu system and make selections.

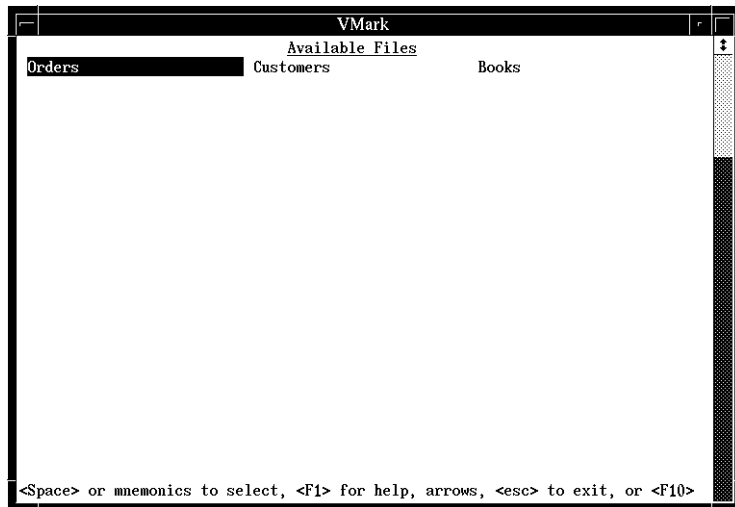
To move the cursor up and down in the submenus, use ↑ and ↓. Note that ↑ does not work on menu bars, and ↓ works only if there is a submenu. To move from a submenu to the one immediately above it, press ←. To return directly to the main menu level from any submenu, press F10.

If you want to exit from any Motif menu or submenu and return to the UniVerse prompt, press Esc.

Examples

This example shows the LIST.FILES menu in Motif format:

>MOTIF LIST.FILES



The next example shows the LIST.FILES menu in standard UniVerse format:

```
>LIST.FILES  
Available Files
```

1. ORDERS
2. CUSTOMERS
3. BOOKS

```
Which would you like? ( 1 - 3 ) ?
```

NLS.ADMIN

Use NLS.ADMIN to invoke the NLS Administration menus. You must be a UniVerse Administrator logged in to the UV account to use NLS.ADMIN.

Syntax

NLS.ADMIN

Description

NLS.ADMIN invokes the NLS Administration menus. From these menus you can:

- Examine the Unicode character set using various search criteria
- View, create, or modify map descriptions and map tables
- View, create, or modify locale definitions
- View, create, or modify category files and weight tables
- Install maps into shared memory or edit the *uvconfig* file

See the *UniVerse NLS Guide* for detailed information about the NLS Administration menus.

NLS.UPDATE.ACCOUNT

Use NLS.UPDATE.ACCOUNT to update the contents of your VOC file in an existing account for UniVerse in NLS mode. NLS.UPDATE.ACCOUNT replaces records in your VOC file with records in the system's NEWACC file.

Syntax

NLS.UPDATE.ACCOUNT

Description

NLS.UPDATE.ACCOUNT updates VOC file entries in existing accounts for UniVerse in NLS mode. New accounts created in NLS mode do not need to be updated.

The command sets the map to NONE for the [&HOLD&](#), [&SAVEDLISTS&](#), and [&PH&](#) files to preserve existing data in those files.

If you run NLS.UPDATE.ACCOUNT in the UV account, and you use SQL tables, you must have DBA privilege to run the command. UniVerse prompts you whether you want to update the SQL catalog. By entering **Y**, you set their maps to NONE.

NLS.UPDATE.ACCOUNT does not change data files in the account. See also UNICODE.FILE for information about converting files.

For more information about the commands and features of NLS, see the *UniVerse NLS Guide*.

Example

The following example updates a user account that existed before the NLS install:

```
>NLS.UPDATE.ACCOUNT
*** Converting file to NLS format: &HOLD&
WARNING: An operating system file will be created with a truncated
name.
Creating file "&UNICODE.000" as Type 30.
Creating file "D_&UNICODE.000" as Type 3, Modulo 1, Separation 2.
Added "@ID", the default record for Retrieve, to "D_&UNICODE.000".
Checking that file's data can be read using DEFAULT NLS map.
```

0 records read.

Converting file's data to UNICODE.

0 records converted.

*** Converting file to NLS format: &SAVEDLISTS&

Checking that file's data can be read using DEFAULT NLS map.

4 records read.

Converting file's data to UNICODE.

4 records converted.

*** Converting file to NLS format: &PH&

Checking that file's data can be read using DEFAULT NLS map.

9 records read.

Converting file's data to UNICODE.

9 records converted.

*** NLS account update finished.

NOTIFY

Use NOTIFY to specify whether to display messages from phantom processes instantly or when UniVerse displays the next system prompt. When you enter UniVerse, NOTIFY OFF is in effect.

Syntax

NOTIFY { ON | OFF }

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
ON	Messages appear immediately.
OFF	Messages appear when UniVerse displays the next system prompt (>).

NOTIFY Parameters

NSELECT

Use NSELECT to create a select list of a subset of data elements contained in an active select list. NSELECT selects only those elements from the active select list that are *not* in a specified file. The subset of elements becomes the currently active select list.

Syntax

NSELECT [DICT] *filename* [FROM *n*] [TO *n*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies a file dictionary.
<i>filename</i>	The name of the file against which NSELECT compares the currently active select list.
FROM <i>n</i>	Uses select list <i>n</i> . <i>n</i> is a number from 0 through 10. The default select list is 0.
TO <i>n</i>	Uses select list <i>n</i> . <i>n</i> is a number from 0 through 10. The default select list is 0.

NSELECT Parameters

Description

When locales are enabled in NLS mode, NSELECT uses the Collate convention of the current locale to determine the collating order for the new list. For more information, see the *UniVerse NLS Guide*.

Example

This example first lists the contents of two files, ONE and TWO:

```
>SORT ONE
SORT ONE 11:07:20am 20 Oct 1995 PAGE 1
ONE.....
```

```
1
2
4
5
6
```

5 records listed.

```
>SORT TWO
SORT TWO 11:07:21am 20 Oct 1995 PAGE 1
TWO.....
```

```
1
2
3
4
```

4 records listed.

Next, a SELECT command creates select list 2, which contains record IDs in file ONE, and an NSELECT command creates select list 1, which contains only record IDs in file TWO that are not in select list 2. The LIST command lists the records in file ONE that are specified by the active select list (select list 1).

```
>SELECT ONE TO 2

5 record(s) selected to SELECT list #2.
>>NSELECT TWO FROM 2 TO 1

2 record(s) selected to SELECT list #1.
>>LIST ONE FROM 1
LIST ONE FROM 1 11:07:22am 20 Oct 1995 PAGE 1
ONE.....
```

```
6
5
```

2 records listed.

OFF

Use OFF to log out of the UniVerse environment or to kill a phantom process.

Syntax

OFF [*-pid#*]

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>-pid#</i>	The process number of a phantom process to log out.

OFF Parameter

Description

When you log out, UniVerse closes all files that you have been using, releases any devices assigned to your terminal, and discards the sentence stack if the VOC file STACKWRITE entry is OFF.

The login name for the *-pid#* qualifier must be the same as your login name. For more information about phantom processes, see the [PHANTOM](#) command.

UNIX System V

OFF returns you to the system login prompt. When you use OFF as a verb, OFF is a synonym for the LOGOUT command.

UNIX Berkeley.

OFF works like QUIT.

Windows Platforms

The behavior depends on how you accessed UniVerse. If you accessed UniVerse from a Windows command window, OFF terminates the UniVerse session and returns control to the command window. If you accessed UniVerse from a telnet interface, OFF terminates UniVerse and the telnet session.

ON.ABORT

Use ON.ABORT to execute a stored sequence of commands when a program aborts.

Syntax

ON.ABORT

Description

The ON.ABORT record in the VOC file executes when a program aborts. An ON.ABORT entry must be a sentence, a paragraph, a menu, a proc, a BASIC program, a verb, or a remote pointer to one of these.

When you first create your UniVerse account, the VOC file does not contain an ON.ABORT entry. You can create or change an ON.ABORT entry in the VOC file at any time. If there is no ON.ABORT entry in the VOC file, you are returned to the UniVerse prompt when a program aborts.

You can use the ON.ABORT record to ensure that certain commands are processed after a fatal error before you are returned to the UniVerse prompt or to rerun the application in which the error occurred.

Example

This example illustrates an ON.ABORT paragraph that alerts you to contact your UniVerse administrator to prevent you from returning to the UniVerse prompt if your application aborts:

```
1          ON.ABORT
001 PA
002 DISPLAY You have aborted from the application.
003 DISPLAY Please contact the system administrator.
004 IF <<Enter Q to log off completely or any other key to
continue>>
= "Q" THEN GO END
005 LOGIN          * RUN LOGIN PARAGRAPH
006 GO EXITPA
007 END: LOGOUT
008 EXITPA:
```

ON.EXIT

Use ON.EXIT to execute a stored sequence of commands when you exit UniVerse.

Syntax

ON.EXIT

Description

The ON.EXIT entry in the VOC file executes when you exit UniVerse. An ON.EXIT entry must be a sentence, a paragraph, a menu, a proc, a BASIC program, a verb, or a remote pointer to one of these.

When you first create your UniVerse account, the VOC file does not contain an ON.EXIT entry. You can create or change an ON.EXIT entry in the VOC file at any time. If there is no ON.EXIT entry in the VOC file, you exit and return to where you invoked UniVerse, or log out altogether, depending on how your UniVerse environment is configured.

You can use the ON.EXIT record to ensure that certain commands are processed before you leave UniVerse or to display the most recently updated records in a file.

Example

This example automatically starts the ON.EXIT paragraph when you exit UniVerse. The paragraph resets the line printer, prints the most recent accounting reports, and displays the reports in the spool queue.

```
1          ON.EXIT
001 PA
002 SETPTR ,,,,3, BRIEF
003 SELECT &HOLD& WITH @ID LIKE "ACCT..."
004 SPOOL &HOLD&
005 SPOOL -LIST
```

P.ATT

Use P.ATT to request control of a printer.

Syntax

P.ATT *n* [–WAIT]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>n</i>	The number of the printer you want to control.
–WAIT	Queues your request for control, waiting until the user who currently controls the printer logs out or releases control of the printer with P.DET .

P.ATT Parameters

Description

If someone else is using the printer you want to attach, the following message appears:

DEVICE IN USE

If that happens, use the –WAIT option to avoid repeating the command while someone else is using the printer.

If you try to run a process requiring control of the printer and you do not use P.ATT, the following message appears:

DEVICE NOT ASSIGNED

P.ATT is a BASIC program in the UV account’s BP file. P.ATT executes an [ASSIGN](#) command with the proper arguments.

P.DET

Use P.DET to release the printer that you attached to your control using [P.ATT](#). This lets other users assign the printer for their use. Enter **P . DET** at the same terminal where you entered the P.ATT command.

Syntax

P.DET *n*

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>n</i>	The number of the printer specified by a previous P.ATT command.

P.DET Parameter

Description

When you log out, UniVerse automatically releases any devices, including printers, that have been assigned to you.

PAGE.MESSAGE

Use PAGE.MESSAGE to suppress the message Press any key to continue . . . at the end of each page of reports that extend beyond one page.

Syntax

PAGE.MESSAGE [ON | OFF]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
ON	Turns on page message.
OFF	Turns off page message.

PAGE.MESSAGE Parameters

Description

The default state is to display the message. Disabling the message suppresses it and makes one extra line available at the bottom of each page. The cursor goes to the end of the last printed line, and no message appears. Output does not scroll beyond each page.

PAGE.MESSAGE with no arguments displays the current state.

PASSWD

Use PASSWD to set or change the password for your current login name.

Syntax

PASSWD [*name*]

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>name</i>	A login ID. If you do not specify <i>name</i> , your login ID is used by default. You must be aUniVerse Administrator to change a password you do not own.

PASSWD Parameter

Description

If you do not enter your old password correctly, the message `Sorry` appears.

If you do not enter the new password the same way twice, PASSWD displays the following message:

Mismatch - password unchanged.

Example

```
>PASSWD
Changing password for jones
Old password:
New password:
Retype new password:
```

PHANTOM

Use PHANTOM to start a process that executes in the background. A phantom process cannot require input from the terminal.

Syntax

PHANTOM [BRIEF] [SQUAWK] *command*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
BRIEF	Directs output from the phantom process to the null device. Messages are not suppressed by the BRIEF option.
SQUAWK	Displays the record ID of the record created in the &PH& file by the phantom process.
<i>command</i>	The command text. The command can be a stored sentence or paragraph or a UniVerse command. <i>command</i> can be up to 25 characters.

PHANTOM Parameters

Description

There is a systemwide limit on the number of processes that you can initiate. If you can start no more processes when you issue a PHANTOM command, the following message appears:

```
NO FREE PHANTOMS
You cannot run a phantom process now.  Wait a while, then try again.
```

If the space is available, the process begins and displays a message similar to the following:

```
Phantom process started as Process ID pid#.
```

The operating system assigns the process ID number, *pid#*.

A phantom process cannot use any terminal services. Use DATA statements in a paragraph to provide input to a phantom process. For example, the following paragraph runs the BASIC program MYPROG and supplies input to it using two DATA statements:

```
                BACKGROUND
0001: PA
0002: RUN BP MYPROG
0003: DATA A
0003: DATA B
```

To run MYPROG as a phantom process, enter the following at the system prompt:

```
>PHANTOM BACKGROUND
```

If a process issues a request for input that is not satisfied by DATA statements, UniVerse logs out the process.

Output from phantom processes is stored in the type 1 file [&PH&](#). Each phantom process creates a record in the [&PH&](#) file with a record ID in the following format:

```
phantom.verb_time_date
```

phantom.verb is the first word in *command* and *time_date* is a unique identifier based on the time and date the process was started. UniVerse directs terminal output to this record. If a phantom process does not produce any output, it creates an empty record. Delete an [&PH&](#) file record when you no longer need it, so the [&PH&](#) file does not become too large. (For more information, see [&PH&](#).)

You can display the output of a phantom process from your terminal. For more information on displaying another user's output, see the [TANDEM](#) command.

Use STATUS ME to monitor the phantom process. The phantom process has the same user ID as your own.

If you are logged in when the phantom process finishes, UniVerse notifies you. If NOTIFY ON is set, the message appears on your screen immediately. Otherwise the message appears when the next UniVerse prompt (>) appears. See [NOTIFY](#) for more information.

Use the [LOGOUT](#) command to stop a phantom process from the same terminal where you started it. A UniVerse Administrator can log out any phantom process on the system.

To make a phantom job the same as a job run from the command line, a phantom process runs the LOGIN paragraph or proc when it starts up.

When a phantom process runs your LOGIN paragraph which in turn runs a menu program, undesirable results can occur (when, for example, the phantom process produces a report).

You can structure your LOGIN paragraph to avoid these problems. Put commands to be executed by regular users and phantoms in the top section of the LOGIN paragraph. Follow this section with a command line that exits the LOGIN paragraph if it is being executed by a phantom process.

To check whether a phantom process is running the LOGIN paragraph, put the following line near the beginning of the paragraph:

```
IF @TTY = 'phantom' THEN GO END.OF.LOGIN
```

Put commands that phantoms should not execute in the last section of the LOGIN paragraph. The last line of the LOGIN paragraph must be the following label:

```
END.OF.LOGIN:
```

PORT.STATUS

Use PORT.STATUS to list currently active UniVerse jobs on the system. You can execute the PORT.STATUS command only from the UV account.

Syntax

PORT.STATUS [*USER name*] [*PORT number*] [*DEVICE pathname*]
[*PID process#*] [FILEMAP] [LAYER.STACK] [MFILE.HIST]
[LOCK.HIST] [{ ENABLE | DISABLE } LOCK.HIST]
[ODBC.CONNECTIONS] [LPTR] [RUNNING]
[SUSPENDED]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
USER	Limits the report to jobs owned by user <i>name</i> . Windows Platforms. This is the user name, with or without the domain name; for example, <i>ibm\jim</i> and <i>jim</i> are both acceptable.
PORT	Limits the report to jobs running on port <i>number</i> .
DEVICE	Limits the report to jobs attached to the device specified by <i>pathname</i> . Windows platforms do not support this option.
PID	Limits the report to the process specified by <i>process#</i> .
LOCK.HIST	Lists concurrency control operations.
ENABLE LOCK.HIST	Enables logging of concurrency control operations.
DISABLE LOCK.HIST	Disables logging of concurrency control operations.

PORT.STATUS Parameters

Parameter	Description
LPTR	Sends the report to the printer.
RUNNING	Lists all UniVerse processes that are not yet suspended.
SUSPENDED	Lists all suspended UniVerse processes.

PORT.STATUS Parameters (Continued)

The following options require that only one session be specified:

Option	Description
FILEMAP	Lists all files currently being used by the specified job.
LAYER.STACK	Generates a process layer stack dump for the indicated job.
MFILE.HIST	Lists the most recent log of rotating file pool (MFILE) operations.
LOCK.HIST	Lists the last 50 concurrency control operations. Logging must previously be enabled (use the ENABLE LOCK.HIST option) in order for LOCK.HIST to work.
ODBC.CONNECTIONS	Lists the data source name, the DBMS type, and the network node name for the connection between the UniVerse system and the server.

PORT.STATUS Options

The PORT.STATUS report shows an R (running) or an S (suspended) in front of the process ID of each process listed. The RUNNING and SUSPENDED keywords limit the PORT.STATUS report to just those processes that are still running or that have been suspended, respectively.

Examples

The PORT.STATUS command with no options produces a report that looks like this:

```
>PORT.STATUS
```

```
There are currently 5 UniVerse sessions; 5 interactive, 0 phantom
```

Pid..	User name.	Who	Port name..	Last command processed.....
6002	walter	13	/dev/ttyp2	PORT.STATUS
5047	cac	16	/dev/ttyp8	LIST CONTACTS <<I2,Enter vendor id>>
5812	barbara	26	/dev/ttypd	SELECT IB WITH DESC LIKE '...LOCATE...'
5844	tamara	47	/dev/ttyq8	LIST IB LONG.DESC FIXED.AT.ANY
5822	tamara	50	/dev/ttyq7	LIST IB LONG.DESC FIXED.AT.ANY

UNIX

The “Who” column lists the port number, and the “Port name” column lists the device pathname of the session.

Windows Platforms. The “Who” column is not displayed, as it the same as the terminal number, and the “User name” column lists the domain and the user name.

The next example uses PORT.STATUS with the LOCK.HIST option to list the last concurrency control operations for the user *philippe*. The “File,” “Group,” and “Record Lock” columns list the type of lock (the Lmode code from the [LIST.READU](#) display). The “Group” column lists the logical disk address of the group, or its offset in bytes from the start of the file. It corresponds to the G-Address from the LIST.READU display.

```
>PORT.STATUS USER philippe LOCK.HIST
```

```
There are currently 1 UniVerse sessions; 1 interactive, 0 phantom
```

Pid..	User name.	Who	Port name.....	Last command processed.....
18468	philippe	26	/dev/pts/34	PORT.STATUS USER philippe LOCK.HIST

File name.....	File	No	Chan
VOC		4	5
SYS.MESSAGE		1	4
SYS.MESSAGE		17	7
VOC		8	6

Lock operation	File No	Group...	File	Group	Record	Lock	Lock	Lock	Caller...
DBrsemv	4	0x00008400				RD			DBread
DBrsemp	4	0x00008400				RD			DBread

The next example uses the PORT.STATUS command with the ODBC.CONNECTIONS option to list information about a BASIC SQL Client Interface connection established for process number 5333:

```
>PORT.STATUS PID 5333 ODBC.CONNECTIONS
There are currently 1 UniVerse sessions; 1 interactive, 0 phantom
Pid.. User name. Who Port name...Last command processed.....
 5333 enf          76 /dev/pts/62 RUN BP LOCALTEST rempubs enf wolfman0
/dbms/enf/newsqico [ LOCALTEST @ 0x41C ]
Data source.....DBMS.....Node.....
rempubs                                UNIVERSE      pubs
```

The “Data source” column lists the name defined in the *uvodbc.config* file, the “DBMS” column lists the type of database used to access the data, and the “Node” column lists the server the DBMS resides on.

Windows Platforms

Details for the current process (this user) are always shown as “Unavailable”.

PRIME

Use PRIME to find the prime numbers closest to a given number.

Syntax

PRIME *n*

Parameter

The following table describes the parameter of the syntax.,

Parameter	Description
<i>n</i>	The number you want to examine.

PRIME Parameter

Description

PRIME returns two numbers, the next highest prime and the next lowest prime. This is useful for determining what number to use as the modulo for a file. Using a prime number for the modulo minimizes hashing conflicts and distributes records evenly.

Example

```
>PRIME 45
Next lower prime number: 43.
Next higher prime number: 47.
```



PRINT.ADMIN

Use PRINT.ADMIN to list the status of printers and spooler queues, to change print job characteristics (such as number of copies to print, which printer or form to use, and so on), and to change the status of print jobs (for example, kill a job, suspend and resume a print job, and so on).

***Note:** This command is not supported on Windows platforms.*

Syntax

PRINT.ADMIN

Description

PRINT.ADMIN invokes the Printer Administration menu. The menu comprises three submenus, one to list printer and spooler queue status, one to control print jobs, and one to change the characteristics of print jobs.

PTERM (UNIX)

Use PTERM to set and display terminal characteristics. If you use PTERM with no options, PTERM lists the options on your terminal.

Syntax

PTERM [LPTR | MTU [*channel*]] [DISPLAY] [*option setting*] ...

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
LPTR	Specifies the terminal characteristics of a line printer channel. <i>channel</i> can be a number from 0 through 225. The default is 0.
MTU	Specifies the terminal characteristics of a magnetic tape channel. <i>channel</i> can be a number from 0 through 7. The default is 0.
DISPLAY	Displays a list of the current terminal characteristics.
<i>option</i>	The terminal characteristics you want to set.
<i>setting</i>	The new setting for the option.

PTERM Parameters

options can be any of the following. Each option includes a list of possible settings.

Option	Description
BAUD	Sets the baud rate of your terminal.
<i>n</i>	The rate to be set. It can be 50, 75, 110, 134, 150, 200, 300, 600, 1200, 1800, 2400, 4800, 9600, EXTA, or EXTB. Zero (0) hangs up the dataset connection.
BGSTOP	Stops a background job if it tries to send output to the terminal.

PTERM Options

Option	Description
BREAK	ON Stops any job running in the background if it attempts to output to your terminal.
	OFF Causes background terminal output to be multiplexed with the foreground output.
	Causes the Intr (interrupt), Quit, Susp (suspend), and Dsusp (delayed-suspend) keys to cause a BREAK condition within UniVerse. Berkeley systems support this by setting the above keys to their default values.
	ON The Intr, Quit, Susp, and Dsusp (delayed-suspend) keys cause a BREAK condition.
	OFF Treats the Intr, Quit, Susp, and Dsusp keys as normal input characters.
	INTR Treats the Break key as Intr.
BRK	IGNORE Ignores the Break key, no interrupt is generated.
	NUL The Break key inputs a NUL (ASCII 0) character.
	Can be used by special programs that require input to terminate on a character other than a newline. UniVerse treats the BRK character as a newline.
	ON Sets the BRK character to Return (Ctrl-M).
BSDELAY	OFF Turns off the BRK character.
	<i>c</i> Sets the BRK character to the character <i>c</i> .
	Sets the backspace delay.
CASE	ON Specifies a delay of about .05 seconds whenever a backspace is output.
	OFF Specifies no delay upon output of a backspace.
	Performs various conversions on uppercase and lowercase characters.
	INVERT Converts uppercase characters to lowercase and lowercase characters to uppercase on input.

PTERM Options (Continued)

Option	Description
NOINVERT	No case conversions occur.
UC-IN	Translates uppercase input to lowercase (which might be inverted to uppercase)
LC-IN	No translation of case on input (even though it still might be inverted).
UC-OUT	Translates lowercase output to uppercase.
LC-OUT	No translation of case on output.
XCASE	Precedes uppercase output by a backslash (\) to distinguish it from lowercase. This is useful when UC-OUT is set because, in that setting, upper- and lowercase are printed as uppercase.
NOXCASE	No distinction on output between upper- and lowercase.
UC	Translates uppercase input to lowercase (which might be inverted to uppercase) and lowercase output to uppercase. On Berkeley systems, this option also sets XCASE.
LC	No translation of case on either input (even though it still might be inverted) or output. On Berkeley systems, this option also sets NOXCASE.
CRMODE	Sets the carriage-return mode.
INLCR	Converts newline to carriage return on input.
NOINLCR	Prohibits conversion of newline to carriage return on input.
IGNCR	Ignores carriage return on input.
NOIGNCR	Does not ignore carriage return on input.
ICRNL	Converts carriage return to newline on input.
NOICRNL	Prohibits conversion of carriage return to newline on input.

PTERM Options (Continued)

Option	Description
	ONLCR Converts newline to newline, carriage return on output.
	NOONLCR Prohibits conversion of newline to newline, carriage return on output.
	OCRNL Converts carriage return to newline on output.
	NOOCRNL Prohibits conversion of carriage return to newline on output.
	ONOCR Prohibits output of carriage return when cursor is in column 0.
	NOONOCR Outputs carriage return when cursor is in column 0.
	ONLRET Newline performs carriage return function.
	NOONLRET Newline does not perform carriage return function.
	T
	ON Sets ICRNL and ONLCR, and resets all other values.
	OFF Resets all CRMODE values.
DATABITS	Changes the number of data bits on the terminal line protocol. Settings: 5, 6, 7, 8.
DSUSP	Sets the DSUSP (delayed-suspend) character. This character acts like the SUSP (suspend) character, except no action is taken until the process actually inputs the character. In this way, the DSUSP is a way to type-ahead a SUSP.
	ON Sets the DSUSP character to Ctrl-Y.
	OFF Turns off the DSUSP character.
	<i>c</i> Sets the DSUSP character to the character <i>c</i> .
DTR	Turns Data Terminal Ready handshaking on or off.
	ON Turns on Data Terminal Ready handshaking. Loss of DTR is treated the same as the input of the STOP character.

PTERM Options (Continued)

Option	Description	
ECHO	OFF	Turns off Data Terminal Ready handshaking. Loss of DTR is treated as a hang-up, and foreground jobs are terminated.
	Sets terminal echo characteristics.	
	ON	Turns the terminal echo on.
	OFF	Turns the terminal echo off.
	FAST	Erases deleted characters from the screen.
	MEDIUM	Erases deleted characters from the screen. The KILL character causes a new line in this mode.
	SLOW	Backspaces over the deleted characters, but does not erase their images from the screen. The KILL character causes a newline in this mode.
	PRINTER	Used for printing terminals so that deleted characters echo backward within \ and /.
	CTRL	Echoes all control characters as ^ followed by the appropriate alphabetic character.
	NOCTRL	Echoes all control characters as nonprintable control characters.
EOF	LF	Echoes the newline character even when the echo is turned off. This mode is useful for some half-duplex terminals.
	NOLF	The newline character does not echo when the echo is turned off.
	Sets the end-of-file mark character. This character terminates input to many UNIX commands (<i>mail</i> , <i>dc</i> , and so forth). UniVerse treats the EOF character as a newline.	
	ON	Sets the end-of-file mark to character Ctrl-D.
	OFF	Turns off the end-of-file mark character.
	<i>c</i>	Sets the end-of-file mark character to the character <i>c</i> .

PTERM Options (Continued)

Option	Description
EOL	The System V UNIX equivalent of the Berkeley UNIX BRK character; its uses are the same. UniVerse treats the EOL character as a newline. ON Sets the end-of-line mark character to Return (Ctrl-M). OFF Turns off the end-of-line mark character. <i>c</i> Sets the end-of-line mark character to the character <i>c</i>.
EOL2	Sets a second end-of-line mark character. ON Sets the second end-of-line mark character to Esc (Ctrl-[]). OFF Turns off the second end-of-line mark character. <i>c</i> Sets the second end-of-line mark character to the character <i>c</i>.
ERASE	Sets the character-erase character. The character-erase character deletes the previous character from the input. ON Sets the character-erase character to Backspace (Ctrl-H). OFF Turns off the character-erase character. <i>c</i> Sets the character-erase character to the character <i>c</i>.
FFDELAY	Sets the delay before formfeed. 0 Pages are sent to the terminal and the line printer, but does not print the clear-screen at the beginning of each page on the terminal, and sends no formfeeds to the line printer. 1 Sends clear-screens to the terminal, but does not send formfeeds to the line printer. 2 Sends clear-screens to the terminal and formfeeds to the line printer; output of a formfeed causes no delay.
PTERM Options (Continued)	

Option	Description
	3 Sends clear-screens to the terminal and formfeeds to the line printer; output of a formfeed causes a two-second delay.
FILL	Causes all delays (FFDELAY, LFDELAY, BSDELAY, TABS, VTDELAY) to pause; they do not use fill characters.
	ON All delays use fill characters. The fill character can be either a NUL or a DEL (see below).
	OFF All delays pause; they do not use fill characters.
	NUL When delays use fill characters, use the NUL character.
	DEL When delays use fill characters, use the DEL character.
FLUSH	Sets the FLUSH character. The FLUSH character stops all output to the terminal. Unlike the STOP character, all output is lost. To resume output, another FLUSH character must be input.
	ON Sets the flush character to Ctrl-O.
	OFF Turns off the flush character.
	<i>c</i> Sets the flush character to the character <i>c</i> .
FMC	Sets the field mark character.
	ON Sets the field mark character to Ctrl-^.
	OFF Turns off the field mark character.
	<i>c</i> Sets the field mark character to the character <i>c</i> .
INBUFF	Determines how the input buffer handles input characters.
	ON Input characters are not transmitted until a carriage return is received. Same as MODE LINE.

PTERM Options (Continued)

Option	Description
	OFF Input characters are transmitted as they are received (like raw mode). The difference from raw mode is that for networking, the data is not packetized until a carriage return is received. Same as MODE EMULATE.
INPUTCTL	Enables or disables input of control characters. ON Allows input of control characters. OFF Disallows input of control characters. TCL.RESET Disallows input of control characters until TCL level is reached.
INTR	Sets the INTR (interrupt) character. The INTR character terminates a currently running job. UniVerse treats the INTR character as a BREAK condition. ON Sets the INTR character to Delete (Ctrl-?). OFF Turns off the INTR character. <i>c</i> Sets the INTR character to the character <i>c</i>.
KILL	Sets the KILL character. The KILL character erases the entire input line. ON Sets the KILL character to Ctrl-X. OFF Turns off the KILL character. <i>c</i> Sets the KILL character to the character <i>c</i>.
LCONT	Sets the line-continue character. The line-continue character is a shorthand way of extending an input line at the Command Language prompt. Typing the LCONT (line-continue) character is the same as entering an underscore (<code>_</code>) followed by a newline. ON Sets the line-continue character to Ctrl-<code>_</code>. OFF Turns off the line-continue character. <i>c</i> Sets the line-continue character to the character <i>c</i>.
LFDELAY	Sets the delay before a linefeed.

PTERM Options (Continued)

Option	Description
	0 Specifies no delay for each newline.
	1 A delay of about .08 seconds occurs after each newline.
	2 A delay of about .10 seconds occurs after each newline.
	3 A delay of about .16 seconds occurs after each newline.
	4 A delay of about .18 seconds occurs after each newline.
	5 A delay of about .26 occurs after each newline.
	6 A delay dependent on the column position occurs after each newline. This mode has been tuned for Teletype model 37s.
	7 A delay dependent on the column position + about .08 seconds occurs after each newline.
	8 A delay dependent on the column position + about .16 seconds occurs after each newline.
LITOUT	Enables or disables postprocessing of output characters.
	ON Outputs characters with normal postprocessing. This setting is the same as <i>opost 0</i> .
	OFF Outputs characters without postprocessing. This setting is the same as <i>opost 1</i> .
LNEXT	Sets the literal-next character. The literal-next character enters literally the next character typed; no input processing occurs. LNEXT (literal-next) can be used to enter the ERASE character literally into text. This option has no effect when used in UniVerse.
	ON Sets the literal-next character to Ctrl-V.
	OFF Turns off the literal-next character.
	<i>c</i> Sets the literal-next character to the character <i>c</i> .

PTERM Options (Continued)

Option	Description
MODE	Determines how the input buffer handles input characters.
LINE	Transmits no input characters until a carriage return is received. Same as INBUFF ON.
RAW	Transmits input characters as they are received.
CHAR	Transmits input characters as they are received, except for special characters.
EMULATE	Input characters are transmitted as they are received (like RAW mode). The difference from RAW mode is that, for networking, the data is not packetized until a carriage return is received. Same as INBUFF OFF.
NOHANG	Sets whether or not the loss of Data Terminal Ready (DTR) is to be ignored.
ON	Ignores loss of Data Terminal Ready; loss of carrier does not terminate a job.
OFF	Treates loss of Data Terminal Ready as a hang up; terminates running jobs.
PARITY	Sets the parity.
NONE	Specifies that no parity generation is done for output, and enforces no parity checking on input.
EVEN	Generates even parity for output and checks for even parity on input (if enabled).
ODD	Generates odd parity for output, and checks for odd parity on input (if enabled).
ENABLE	Enables parity input checking, provided that the parity mode is not set to NONE.
DISABLE	Disables input parity; allows characters of any parity.
ERR-IGN	Ignores errors (characters of the wrong parity) if input parity checking is enabled.

PTERM Options (Continued)

Option	Description
	ERR-MRK Marks errors by simulating a special input sequence, if input parity checking is enables. You cannot use this mode in UniVerse. If set, it acts like ERR-IGN.
	ERR-NUL Inputs errors as the NUL character, if input parity checking is enabled.
PENDIN	Automatically retypes input when background output disturbs the terminal screen.
	ON Automatically retypes input, and enters an ERASE character. This mode has no effect in UniVerse.
	OFF No automatic retyping of input.
QUIT	Sets the QUIT character. The QUIT character terminates a currently running job and produces a core dump. UniVerse treats the QUIT character as a BREAK condition.
	ON Sets the QUIT character to Ctrl-\..
	OFF Turns off the QUIT character.
	<i>c</i> Sets the QUIT character to the character <i>c</i> .
RPRNT	Sets the reprint character. The reprint charcter redisplayes the previous line. This is often useful when transmission errors or background output disturb the data on the terminal screen.
	ON Sets the reprint character to Ctrl-R.
	OFF Turns off the reprint character.
	<i>c</i> Sets the reprint character to the character <i>c</i> .
SMC	Sets the subvalue mark character
	ON Sets the subvalue mark character to Ctrl-\..
	OFF Turns off the subvalue mark character.
	<i>c</i> Sets the subvalue mark character to the character <i>c</i> .
SQLNULL	Sets the SQL null value character.
	ON Sets the null value character to Ctrl-N.

PTERM Options (Continued)

Option	Description
START	OFF Turns off the null value character.
	<i>c</i> Sets the null value character to the character <i>c</i> .
	Sets the START character. The START character is the counterpart of the STOP character. START resumes output after it has stopped. If the XON STARTANY option is set, any input character resumes output, and the START character is the only character not entered as data.
	ON Sets the START character to Ctrl-Q.
STOP	OFF Turns off the START character.
	<i>c</i> Sets the START character to the character <i>c</i> .
	Sets the STOP character. The STOP character temporarily stops output to the terminal screen. Typing the START character resumes printing of output.
	ON Sets the STOP character to Ctrl-S.
STOPBITS	OFF Turns off the STOP character.
	<i>c</i> Sets the STOP character to the character <i>c</i> .
	Sets the number of stop bits on the terminal line protocol.
	1 Sets the terminal line protocol for 1 stop bit.
STRIP	2 Sets the terminal line protocol for 2 stop bits.
	Determines whether or not to strip the eighth bit off input character.
	ON Strips off the eighth bit.
	OFF Does not strip off the eighth bit.
SUSP	Sets the SUSP (suspend) character. The SUSP character immediately stops the current job. UniVerse treats the SUSP character as a BREAK condition.
	ON Sets the SUSP character to Ctrl-Z.
	OFF Turns off the SUSP character.

PTERM Options (Continued)

Option	Description
SWTCH	<i>c</i> Sets the SUSP character to the character <i>c</i> .
	Sets the SWTCH character. The SWTCH character is used in conjunction with <i>shl</i> to switch terminal input to the layering program (<i>shl</i>).
	ON Sets the SWTCH character to Ctrl-Z.
	OFF Turns off the SWTCH character.
TABS	<i>c</i> Sets the SWTCH character to the character <i>c</i> .
	Expands a tab character on output to the proper number of spaces. Tab stops are set every eight columns. Some terminals (like the ADDS Viewpoint) use a tab character as a part of the cursor movement function. On these terminals, TABS must be set to OFF for cursor movement to work properly.
	ON Turns on tab expansion.
	OFF Turns off tab expansion
TILDE	Enables or disables conversion of the tilde (~) character.
	ON Converts ~ to ‘ (accent grave) on output.
	OFF Does not convert ~.
TMC	Sets the text mark character.
	ON Sets the text mark character to Ctrl-T.
	OFF Turns off the text mark character.
	<i>c</i> Sets the text mark character to the character <i>c</i> .
VMC	Sets the value mark character.
	ON Sets the value mark character to Ctrl-J.
	OFF Turns off the value mark character.
	<i>c</i> Sets the value mark character to the character <i>c</i> .
VTDELAY	Sets the vertical tab delay.

PTERM Options (Continued)

Option	Description
WERASE	ON Specifies a two-second delay each time a vertical tab is output.
	OFF Specifies no delay time when a vertical tab is output.
	Sets the word-erase character. The word-erase character deletes the previous word (up to, but not including, a space).
	ON Sets the word-erase character to Ctrl-W.
	OFF Turns off the word-erase character.
XON	<i>c</i> Sets the word-erase character to the character <i>c</i> .
	Sets the X-ON and the X-OFF character.
	Berkeley Systems: The X-OFF character is the STOP character and the Y-ON character is the START character. This option is implemented by setting STOP and START to their default values.
	UNIX System V: The X-OFF character is always Ctrl-S, and the X-ON is always Ctrl-Q.
	ON Turns on the X-OFF/X-ON protocol. When the computer receives an X-OFF, it stops all transmission until it receives an X-ON.
STARTAN Y	OFF Turns off the X-OFF/X-ON protocol. The X-OFF and the X-ON characters are treated as normal input. Berkeley systems implement this option by turning off the STOP and the START characters.
	Any character acts as X-ON, if X-OFF/X-ON is enabled.

PTERM Options (Continued)

Option	Description
NOSTART ANY	Requires the receipt of the X-OFF character to restart the transmission.
TANDEM	Causes the computer, when its input buffer is almost full, to transmit an X-OFF character to the terminal screen, and when the buffer is almost empty, to transmit an X-ON. This lets the computer communicate with another device, such as another computer.
NOTANDE M	Turns off the automatic X-OFF/X-ON mode.

PTERM Options (Continued)

Description

Berkeley systems do not support the following options:

DATABITS
EOL
EOL2
STOPBITS
STRIP
SWTCH
VTDELAY

UNIX System V

These systems do not support the following options:

BGSTOP
BRK
DSUSP
DTR
FLUSH
LNEXT
NOHANG
PENDIN
SUSP
TILDE

When specifying output-processing *stty* parameters such as ONLCR in a PTERM command or in the *sp.config* file, you must also specify –LITOUT to get the postprocessing done.

For further discussion of terminal characteristics and settings, see the *stty* command in your UNIX documentation.

NLS Mode

PTERM ECHO CTRL may produce unexpected results for non-ASCII control characters. In addition, PTERM CASE INVERT does not affect characters entered through deadkey sequences. For more information about characters, see the *UniVerse NLS Guide*.

Examples

To display a list of PTERM options, enter the following:

```
>PTERM
PTERM options are:
ERASE      char | ON | OFF      KILL       char | ON | OFF
WERASE     char | ON | OFF      RPRNT     char | ON | OFF
INTR       char | ON | OFF      QUIT      char | ON | OFF
EOF        char | ON | OFF      EOL       char | ON | OFF
EOL2       char | ON | OFF      LCONT     char | ON | OFF
FMC        char | ON | OFF      VMC       char | ON | OFF
SMC        char | ON | OFF

BAUD       50 - 9600
BREAK
ON | OFF | INTR | IGNORE | NUL
BSDELAY     ON | OFF
CASE       INVERT | NOINVERT | LC-IN | LC-OUT | UC-IN | UC-OUT | XCASE |
NOXCASE
CRMODE     INLCR | -INLCR | IGNCR | -IGNCR | ICRNL | -OICRNL |
          ONLCR | -ONLCR | OCRNL | -CRNL | ONLRET | -ONLRET |
          ON | OFF
DATABITS   5 - 8
ECHO       ON | OFF | CTRL | NOCTRL | FAST | MEDIUM | SLOW | LF | NOLF
FFDELAY    0 - 3
FILL       ON | OFF | NUL | DEL
LFDELAY    0 - 8
PARITY     EVEN | ODD | NONE | ENABLE | DISABLE | ERR_IGN | ERR_MRK | ERR_NUL
STOPBITS   1 - 2
TABS       ON | OFF
VTDELAY    ON | OFF
XON        ON | OFF | STARTANY | NOSTARTANY | TANDEM | NOTANDEM
```

To display the current terminal settings, enter the following:

```
>PTERM DISPLAY
MODE          EMULATE
CC            INTR   = ^C   QUIT   = ^_   SUSP   = OFF  DSUSP  = OFF
              SWITCH = ^@   ERASE  = ^H   WERASE = OFF  KILL   = ^U
              LNEXT  = OFF  REPRINT = OFF  EOF    = ^D   EOL    = ^@
              EOL2   = ^@   FLUSH  = OFF  START  = ^Q   STOP   = ^S
              LCONT  = ^_   FMC    = ^^   VMC    = ^]   SMC    = ^\
              TMC    = ^T   SQLNULL = ^N
CARRIER      -RECEIVE -HANGUP -LOCAL
CASE          -UCIN -UCOUT -XCASE -INVERT
CRMODE        -INLCR -IGNCR -ICRNL -ONLCR -OCRNL -ONOCR -ONLRET
DELAY         BSO CR0 FF0 LF0 VT0 TAB0 -FILL
ECHO          ECHO -ERASE -KILL -CTRL -LF
HANDSHAKE     -XON -ANY -TANDEM -DTR
OUTPUT        -POST -TILDE -BG -CS -EXPAND
PROTOCOL      LINE=0 BAUD=0 DATA=0 STOP=0 NONE DISABLE -STRIP
SIGNALS       -ENABLE -FLUSH BREAK=IGNORE
```

To set the ECHO parameter to SLOW, enter the following:

```
>PTERM ECHO SLOW
```

PTERM (Windows Platforms)

Use PTERM to set and display terminal characteristics. If you use PTERM with no options, PTERM lists the options on your terminal.

Syntax

PTERM [LPTR *channel*] [DEVICE *name*] [DISPLAY] [*option setting*] ...

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
LPTR	Specifies the terminal characteristics of a line printer channel. <i>channel</i> can be a number from 0 through 225. The default is 0.
DEVICE	Specifies the device characteristics of a device. <i>name</i> is the name of the device as specified in the &DEVICE& file.
DISPLAY	Displays a list of the current terminal characteristics.
<i>option</i>	The terminal characteristic you want to set.
<i>setting</i>	The new setting for the option.

PTERM Parameters

options can be any of the following. Each option includes a list of possible settings.

Option	Description
BAUD	Sets the baud rate of your terminal. n The rate to be set. It can be 50, 75, 110, 134, 150, 200, 300, 600, 1200, 1800, 2400, 4800, 9600, EXTA, or EXTB. Zero (0) hangs up the dataset connection.
CASE	Performs various conversions on uppercase and lowercase characters.

PTERM Options

Option	Description
CRMODE	INVERT Converts uppercase characters to lowercase and lowercase characters to uppercase on input.
	NOINVERT No case conversions occur.
	Sets the carriage-return mode.
	ICRNL Converts carriage return to newline on input.
ECHO	NOICRNL Prohibits conversion of carriage return to newline on input.
	Sets terminal echo characteristics.
	ON Turns the terminal echo on.
	OFF Turns the terminal echo off.
	CTRL Echoes all control characters as ^ followed by the appropriate alphabetic character.
	NOCTRL Echoes all control characters as nonprintable control characters. This option is not supported in DOC command windows.
	LF Echoes the newline character even when the echo is turned off. This mode is useful for some half-duplex terminals.
	NOLF The newline character does not echo when the echo is turned off.
ERASE	Sets the character-erase character. The character-erase character deletes the previous character from the input.
	ON Sets the character-erase character to Backspace (Ctrl-H).
	OFF Turns off the character-erase character.
	<i>c</i> Sets the character-erase character to the character <i>c</i> .
FMC	Sets the field mark character.
	ON Sets the field mark character to Ctrl-^.

PTERM Options (Continued)

Option	Description
INBUFF	OFF Turns off the field mark character.
	<i>c</i> Sets the field mark character to the character <i>c</i> .
	Determines how the input buffer handles input characters.
	ON Input characters are not transmitted until a carriage return is received. Same as MODE LINE.
	OFF Input characters are transmitted as they are received (like RAW mode). The difference from RAW mode is that, for networking, the data is not packetized until a carriage return is received. Same as MODE EMULATE.
INPUTCTL	Enables or disables input of control characters.
	ON Allows input of control characters.
	OFF Disallows input of control characters.
	TCL.RESET Disallows input of control characters until TCL level is reached.
INPUTMIN	Determines how input of control characters occurs.
	ON Input is not completed until at least one character has been received.
	OFF Input is completed immediately, or after a timeout period, whether or not characters have been received.
INPUTTIME	Sets the timeout value in tenths of a second.
INTR	Sets the INTR (interrupt) character. The INTR character terminates a currently running job. UniVerse treats the INTR character as a BREAK condition.
	ON Sets the INTR character to Delete (Ctrl-?). Not supported from a DOS command window.
	OFF Turns off the INTR character. Not supported from a DOS command window.

PTERM Options (Continued)

Option	Description
KILL	<i>c</i> Sets the INTR character to the character <i>c</i> . Not supported from a DOS command window.
	Sets the KILL character. The KILL character erases the entire input line.
	ON Sets the KILL character to Ctrl-X. Not supported from a DOS command window.
LCONT	OFF Turns off the KILL character. Not supported from a DOS command window.
	<i>c</i> Sets the KILL character to the character <i>c</i> .
	Sets the line-continue character. The line-continue character is a shorthand way of extending an input line at the command language prompt. Typing the LCONT (line-continue) character is the same as entering an underscore (_) followed by a newline.
MODE	ON Sets the line-continue character to Ctrl-_.
	OFF Turns off the line-continue character.
	<i>c</i> Sets the line-continue character to the character <i>c</i> .
	Determines how the input buffer handles input characters.
	RAW Transmits input characters as they are received. This is the equivalent of setting INPUTMIN on and INPUTTIME to 0.
PARITY	CHAR Transmits input characters as they are received, exempt for special characters.
	EMULATE Input characters are transmitted as they are received (like raw mode). The difference from raw mode is that for networking, the data is not packetized until a carriage return is received. Same as INBUFF OFF.
	LINE Input characters are not transmitted until a carriage return is received.
PARITY	Sets the parity.
	NONE Specifies that no parity generation is done for output, and enforces no parity checking on input.

PTERM Options (Continued)

Option	Description
EVEN	Generate even parity for output and checks for even parity on input (if enabled).
ODD	Generates odd parity for output, and checks for odd parity on input (if enabled).
ENABLE	Requests input parity checking with abort on error, so that any input parity error will cause a READSEQ or READBLK statement to take its ELSE clause.
DISABLE	Disables input parity checking.
ERR-MRK	Requests the marking of parity errors. Any input character with a parity error is preceded by an item mark character (byte value 255).
ERR-NUL	Requests the marking of parity errors. Any input character with a parity error is preceded by a NUL character (byte value 0).
RPRNT	Sets the reprint character. The reprint character redisplay the previous line. This is often useful when transmission errors or background output disturb the data on the terminal screen.
ON	Sets the reprint character to Ctrl-R.
OFF	Turns off the reprint character.
<i>c</i>	Sets the reprint character to the character <i>c</i> .
SMC	Sets the subvalue mark character.
ON	Sets the subvalue mark character to Ctrl-\.
OFF	Turns off the subvalue mark character.
<i>c</i>	Sets the subvalue mark character to the character <i>c</i> .
SQLNULL	Sets the SQL null value character.
ON	Sets the null value character to Ctrl-N.
OFF	Turns off the null value character.
<i>c</i>	Sets the null value character to the character <i>c</i> .

PTERM Options (Continued)

Option	Description
STOPBITS	Sets the number of stop bits on the terminal line protocol.
TMC	Sets the text mark character.
	ON Sets the text mark character to Ctrl-T.
	OFF Turns off the text mark character.
	<i>c</i> Sets the text mark character to the character <i>c</i> .
VMC	Sets the value mark character.
	ON Sets the value mark character to Ctrl-].
	OFF Turns off the value mark character.
	<i>c</i> Sets the value mark character to the character <i>c</i> .
WERASE	Sets the word-erase character. The word-erase character deletes the previous word (up to, but not including, a space).
	ON Sets the word-erase character to Ctrl-W.
	OFF Turns off the word-erase character.
	<i>c</i> Sets the word-erase character to the character <i>c</i> .
XON	Sets the X-ON and X-OFF characters.
	ON Turns on the X-OFF/X-ON protocol. When the computer receives an X-OFF, it stops all transmission until it receives an X-ON.
	OFF Turns off the X-OFF/X-ON protocol. The X-OFF and the X-ON characters are treated as normal input.
	TANDEM Enablex input X-OFF/X-ON processing.
	NOTANDEM Disables input X-OFF/X-ON processing.

PTERM Options (Continued)

Description

The allowable options for the PTERM command depend on the type of device being referenced: a console window, a telnet session, or a serial port. If an option is used for a device that is not supported, an error message is displayed.

NLS Mode

PTERM ECHO CTRL may produce unexpected results for non-ASCII control characters. In addition, PTERM CASE INVERT does not affect characters entered through deadkey sequences. For more information about characters, see the *UniVerse NLS Guide*.

Examples

To display a list of PTERM options, enter the following:

```
>PTERM
PTERM options are:
ERASE      char | ON | OFF      KILL       char | ON | OFF
WERASE     char | ON | OFF      RPRNT      char | ON | OFF
INTR       char | ON | OFF      QUIT       char | ON | OFF
EOF        char | ON | OFF      EOL        char | ON | OFF
EOL2       char | ON | OFF      LCONT      char | ON | OFF
FMC        char | ON | OFF      VMC        char | ON | OFF
SMC        char | ON | OFF
BAUD       50 - 9600
BREAK
ON | OFF | INTR | IGNORE | NUL
BDELAY     ON | OFF
CASE       INVERT | NOINVERT | LC-IN | LC-OUT | UC-IN | UC-OUT |
XCASE | NOXCASE
CRMODE     INLCR | -INLCR | IGNCR | -IGNCR | ICRNL | -OICRNL |
          ONLCR | -ONLCR | OCRNL | -CRNL | ONLRET | -ONLRET |
          ON | OFF
DATABITS   5 - 8
ECHO       ON | OFF | CTRL | NOCTRL | FAST | MEDIUM | SLOW | LF |
NOLF
FFDELAY    0 - 3
FILL       ON | OFF | NUL | DEL
LFDELAY    0 - 8
PARITY     EVEN | ODD | NONE | ENABLE | DISABLE | ERR_IGN | ERR_MRK
| ERR_NUL
STOPBITS   1 - 2
TABS       ON | OFF
VTDELAY    ON | OFF
XON        ON | OFF | STARTANY | NOSTARTANY | TANDEM | NOTANDEM
```

To display the current terminal settings, enter the following:

```
>PTERM DISPLAY
MODE          EMULATE
CC            INTR   = DEC QUIT   = ^\   SUSP   = ^Z   DSUSP  =
^Y
              SWITCH = OFF ERASE  = ^H   WERASE = ^W   KILL   =
^U
              LNEXT  = ^V   REPRINT = ^R   EOF    = ^D   EOL    =
OFF
              EOL2   = OFF FLUSH   = ^O   START  = ^Q   START  =
^S
              LCONT  = ^_   FMC     = ^^   VMC    = ^ ]   SMC    = ^
]
              TMC    = ^T   SQLNULL = ^N
CARRIER      -RECEIVE -HANGUP -LOCAL
CASE          -UCIN  -UCOUT -XCASE -INVERT
CRMODE        -INLCR -IGNCR -ICRNL -ONLCR -OCRNL -ONOCR -ONLRET
DELAY         BS0 CR0 FF0 LF0 VT0 TAB0 -FILL
ECHO          ECHO  -ERASE -KILL -CTRL -LF
HANDSHAKE     -XON  -ANY  -TANDEM -DTR
OUTPUT        -POST -TILDE -BG  -CS  -EXPAND
PROTOCOL      LINE=0 BAUD=0 DATA=0 STOP=0 NONE DISABLE -STRIP
SIGNALS       -ENABLE -FLUSH BREAK=IGNORE
```

To set the ECHO parameter to SLOW, enter the following:

```
>PTERM ECHO SLOW
```

PTIME

Use PTIME to display information about your use of the system during the current terminal session. The nature of the information depends on what system UniVerse is running on (UNIX System V, BSD UNIX, or Windows platforms).

Syntax

PTIME

Examples

BSD UNIX. PTIME produces a report that looks like this:

```
>PTIME
You have been logged in for: 10 minutes 28 seconds

CPU times      :   16.049 user          24.166 system
Page faults    :       279 I/O          165 non I/O          0
swapout
Disk I/O       :       383 reads          877 writes
IPC messages   :         3 sent           4 received          0
signals
Context switch:    1183 voluntary        47 involuntary
>
```

System V UNIX and Windows Platforms. PTIME produces a report that looks like this:

```
>PTIME
You have been logged in for: 20 minutes 11 seconds

CPU times: 0.500 user 0.430 system
>
```

QSELECT

Use QSELECT to create a select list containing values from a specified field or from all fields of selected records.

Syntax

QSELECT [DICT] *filename* [*records* | *] [*options*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
<i>filename</i>	The name of a UniVerse file.
<i>records</i>	A list of record IDs.
*	Specifies all records in the file.

QSELECT Parameters

options can be any of the following:

Option	Description
SAVING <i>n</i>	Specifies a single field (the <i>n</i> th field) from which to extract values from each record.
TO <i>n</i>	Saves the select list in a numbered select list. <i>n</i> specifies the list number from 0 through 10.
FROM <i>n</i>	Supplies the <i>record.list</i> from a numbered select list. <i>n</i> specifies the list number from 0 through 10.

QSELECT Options

Description

You can supply an explicit record ID list on the command line. You can also use an active select list to select records.

QSELECT builds a select list from data it extracts from fields in one or more records in a file. QSELECT extracts data from all fields unless you use the SAVING keyword to specify data from one field. QSELECT stores multivalues as separate elements in the list.

Example

This example creates a select list whose elements come from field 3 of records 4309 and 7575 in the file SUN.MEMBER. The [SAVE.LIST](#) command saves the list under a default record ID, and the [EDIT.LIST](#) command edits the saved list.

```
>QSELECT SUN.MEMBER "4309" "7505" SAVING 3

2 record(s) selected to SELECT list #0.
>>SAVE.LIST

2 record(s) SAVED to SELECT list "&TEMP3&".
>EDIT.LIST &TEMP3&
2 lines long.

----:P
0001:4309
0002:7505
Bottom at line 2.
----:Q
```

QUIT

Use QUIT or Q to exit UniVerse and return to your parent process.

Syntax

Q

QUIT

Description

When you quit, UniVerse closes all files that you have been using, releases any devices that have been assigned to your terminal, and discards the sentence stack if the STACKWRITE entry in the VOC file is OFF.

UNIX

If your UniVerse process is the child of a UNIX shell, the shell prompt appears. If your UniVerse process is not the child of a UNIX shell, the system login prompt appears.

Windows Platforms

The way this command works depends on how you accessed UniVerse. If you accessed UniVerse from a Windows command window, QUIT terminates the UniVerse session and returns control to the command window. If you accessed UniVerse from a telnet interface, QUIT terminates UniVerse and the telnet session.

RADIX

Use RADIX to convert hexadecimal, octal, or binary integers to decimal integers, or to convert decimal integers to hexadecimal, octal, or binary integers. RADIX also lets you add, subtract, multiply, and divide integers in all four number bases.

Syntax

RADIX

Description

RADIX prompts for the type of action with the following message:

XTD	OTD	BTD	
DTX	DTO	DTB	
ADDX	ADDO	ADDB	ADDD
SUBX	SUBO	SUBB	SUBD
MULX	MULO	MULB	MULD
DIVX	DIVO	DIVB	DIVD

Input action code =

Enter one of the codes to specify the action you want. The following message appears:

Input *Number.base* String or New Action Code =

Enter a number or a new action code. RADIX performs the operation and then prompts for another action code.

You can specify the following action codes.

Conversion Codes

The following table describes the RADIX conversion codes.

Code	Description
XTD	Converts from hexadecimal to decimal.
OTD	Converts from octal to decimal.
BTD	Converts from binary to decimal.
DTX	Converts from decimal to hexadecimal.
DTO	Converts from decimal to octal.
DTB	Converts from decimal to binary.
RADIX Conversion Codes	

Addition Codes

The following table describes the RADIX addition codes.

Code	Description
ADDX	Adds hexadecimal numbers.
ADDO	Adds octal numbers.
ADDB	Adds binary numbers.
ADDD	Adds decimal numbers.
RADIX Addition Codes	

Subtraction Codes

The following tables describes the RADIX subtraction codes.

Code	Description
SUBX	Subtracts hexadecimal numbers.
SUBO	Subtracts octal numbers.
SUBB	Subtracts binary numbers.
SUBD	Subtracts decimal numbers.
RADIX Subtraction Codes	

Multiplication Codes

The following table describes the RADIX multiplication codes.

Code	Description
MULX	Multiplies hexadecimal numbers.
MULO	Multiplies octal numbers.
MULB	Multiplies binary numbers.
MULD	Multiplies decimal numbers.
RADIX Multiplication Codes	

Division Codes

The following table describes the RADIX division codes.

Code	Description
DIVX	Divides hexadecimal numbers.
DIVO	Divides octal numbers.
DIVB	Divides binary numbers.
DIVD	Divides decimal numbers.
RADIX Division Codes	

Because ADDD and ADDB signify action codes (add decimal and add binary numbers respectively), you must enter the hex numbers ADDD and ADDB as 0ADDD and 0ADDB.

You can enter any action code at the system prompt as a command. If you also specify a value, RADIX performs the action, displays the answer, and prompts for a new number or action code. If you do not specify a number, RADIX prompts for it.

RADIX works with 32-bit signed numbers. For example, RADIX XTD 7FFFFFFF returns 2147483647. If you add 1 to that, the number overflows and RADIX returns -2147483648. Thus, operations on some very large numbers fail and others do not, depending on whether the result fits into a 32-bit number.

Example

The RADIX command in the following example displays the available action codes. The user enters **XTD** to convert a hexadecimal number to decimal, then enters the hex number **ABB**. Next the user enters **BTD** to convert a binary number to decimal, then enters the binary number **0101**. The user then enters **ADDD** to add the decimal numbers 4 and 12. Pressing ENTER returns the user to the system prompt.

```
>RADIX
      XTD          OTD          BTD
      DTX          DTO          DTB

      ADDX         ADDO         ADDB         ADDD
      SUBX         SUBO         SUBB         SUBD
      MULX         MULO         MULB         MULD
      DIVX         DIVO         DIVB         DIVD
Input action code = XTD

Input Hex String or New Action Code =   ABB
Decimal 2747
Input Hex String or New Action Code =   BTD
Input Binary String or New Action Code =  0101
Decimal 5
Input Binary String or New Action Code =  ADDD
Input Decimal String or New Action Code =  4
Input Decimal String = 12
Decimal 16
Input Decimal String or New Action Code =  <Return>
>
```

RAID

Use RAID to debug BASIC programs.

Syntax

RAID [*filename*] *program* [*options*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>filename</i>	The name of the file where the program is stored. RAID appends .O to <i>filename</i> to locate and operate on the object code. If you do not specify <i>filename</i> , RAID assumes the BP file by default.
<i>program</i>	The name of the record containing the source code of the program.

RAID Parameters

options can be any of the following:

Option	Description
NO.WARN	Suppresses all warning (nonfatal) error messages. If you do not specify NO.WARN, run-time error messages appear on the screen as they are encountered.
NO.PAGE	Turns off automatic paging. Programs that position the cursor with @ functions do not need to disable pagination.
LPTR	Spools the program output to the printer rather than to the terminal screen.
KEEP.COMMON	Maintains the values of variables in unnamed common if program execution is passed to another BASIC program with a CHAIN statement.
TRAP	Enters the RAID debugger instead of displaying warning (nonfatal) error messages.

RAID Options

Description

RAID can function as either an object code or a source code debugger. Thus, it is a powerful tool for detecting errors in UniVerse BASIC code. RAID lets you set and delete breakpoints, set watchpoints, step through and display source code, examine object addresses, and examine and modify variables. Use RAID in the same way you use RUN, to invoke the debugger just before program execution.

When a BASIC program is running, you can enter the RAID debugger by pressing the Break key and then selecting the break option D.

The following table contains the commands that are available in RAID. For syntax and a detailed description of each, see *UniVerse Basic*.

Command	Description
<i>line</i>	Displays the specified line of the source code.
<i>/[string]</i>	Searches the source code for <i>string</i> .
B	Sets a RAID breakpoint.
C	Continues program execution.
D	Deletes a RAID breakpoint.
G	Goes to a line or address, and continues program execution.
H	Displays statistics for the program.
I	Displays and executes the next object code instruction.
L	Displays the next line to be executed.
M	Sets watchpoints.
Q	Quits RAID.
R	Runs the program.
S	Steps through the BASIC source code.
T	Displays the call stack trace.
V	Enters verbose mode for the M command.
V*	Prints the compiler version that generated the object code.

RAID Commands

Command	Description
W	Displays the current window.
X	Displays the current object code instruction and address.
X*	Displays local run machine registers and variables.
Z	Displays the next 10 lines of source code.
\$	Turns on instruction counting.
#	Turns on program timing.
+	Increments the current line.
–	Decrements the current line.
.	Displays object code instruction and address before execution.
<i>variable/</i>	Prints the value of <i>variable</i> .
<i>variable!string</i>	Changes the value of <i>variable</i> to <i>string</i> .

RAID Commands (Continued)

RAID.GUI

Use RAID.GUI to set or change the way RAID displays output.

Syntax

RAID.GUI [ON | OFF | SHOW]

RAID.GUI [START | STOP | END | INPUT] *string*

RAID.GUI [STARTEX | STOPHEX | ENDEX | INPUTHEX] *string*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
ON	Turns on special GUI formatting display for RAID debugging. This encapsulates all RAID output within specified start and stop strings.
OFF	Turns off special formatting. This is the default.
SHOW	Displays the state of RAID GUI mode, and start and stop strings.
START	Specifies the start string.
STOP	Specifies the stop string.
END	Specifies the string to use for the end sequence, when RAID is running.
INPUT	Specifies the string to use when RAID asks for input.
STARTEX	Specifies the start string in hexadecimal format.
STOPHEX	Specifies the stop string in hexadecimal format.
ENDEX	Specifies (in hexadecimal format) the string to use for the end sequence, when RAID is running.
INPUTHEX	Specifies (in hexadecimal format) the string to use when RAID asks for input.

RAID.GUI Parameters

Description

The new GUI for the RAID debugger uses encapsulated RAID output to parse the display and forward messages to the appropriate GUI windows. When you enable GUI display for RAID, all RAID output is encapsulated within specified start and end strings.

Examples

The following example shows how to display the current status of the system:

```
>RAID.GUI SHOW
RAID GUI mode is non-active
RAID GUI Start String = 0x1B[=1D
RAID GUI Stop String  = 0x1B[=2D
RAID GUI End String   = 0x1B[3D
RAID GUI Input String  = 0x1B[=4D
```

This display shows the default values for the start, stop, end, and input strings, and it shows nonprintable characters in hexadecimal format. The Esc character (0x1B) is the first character in all strings, followed by the rest of the sequence.

The next example turns encapsulation on:

```
>RAID.GUI ON
```

The next example changes the start string:

```
>RAID.GUI START [START]
>RAID.GUI SHOW
RAID GUI mode is active
RAID GUI Start String = [START]
RAID GUI Stop String  = 0x1B[=2D
RAID GUI End String   = 0x1B[3D
RAID GUI Input String  = 0x1B[=4D
```

After you specify start and stop strings, RAID output changes. For example, what used to look like this:

```
>RAID BP TEST
TEST:      1: A="TEST"
::
```

now looks like this:

```
>RAID.GUI SHOW
RAID GUI mode is active
RAID GUI Start String = [START]
RAID GUI Stop String  = [STOP]
RAID GUI End String   = 0x1B[3D
RAID GUI Input String  = 0x1B[=4D
>RAID BP TEST
[START] TEST:      1: A= "TEST"
[STOP] [START]::[STOP]
```

REBUILD.DF

Use REBUILD.DF to examine the headers of a distributed file and its part files. REBUILD.DF lists inconsistencies and fixes them.

Syntax

REBUILD.DF *dist.filename algorithm*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>dist.filename</i>	The name of a distributed file.
<i>algorithm</i>	Can be either EXTERNAL, INTERNAL, or SYSTEM. See the DEFINE.DF command for details about the partitioning algorithm.

REBUILD.DF Parameters

Use the following syntax to specify the partitioning algorithm:

Parameter	Description						
INTERNAL	The INTERNAL clause has the following syntax. INTERNAL {<i>[[DATA] filename] record.ID expression</i>} <table><tr><td>DATA</td><td>Specifies that the algorithm expression is stored in the data file of <i>filename</i>. If you do not use DATA, the expression is stored in the dictionary of <i>filename</i>.</td></tr><tr><td><i>filename</i></td><td>The name of the file containing the algorithm expression.</td></tr><tr><td><i>record.ID</i></td><td>The ID of the record containing the algorithm expression. If you do not specify <i>filename</i>, the record is assumed to be an entry in the VOC file.</td></tr></table>	DATA	Specifies that the algorithm expression is stored in the data file of <i>filename</i> . If you do not use DATA, the expression is stored in the dictionary of <i>filename</i> .	<i>filename</i>	The name of the file containing the algorithm expression.	<i>record.ID</i>	The ID of the record containing the algorithm expression. If you do not specify <i>filename</i> , the record is assumed to be an entry in the VOC file.
DATA	Specifies that the algorithm expression is stored in the data file of <i>filename</i> . If you do not use DATA, the expression is stored in the dictionary of <i>filename</i> .						
<i>filename</i>	The name of the file containing the algorithm expression.						
<i>record.ID</i>	The ID of the record containing the algorithm expression. If you do not specify <i>filename</i> , the record is assumed to be an entry in the VOC file.						

Partitioning Algorithm Parameters

Parameter	Description
	<i>expression</i> The I-type expression used as the partitioning algorithm. Enclose <i>expression</i> in quotation marks if it contains spaces.
EXTERNAL	Specifies a routine as the partitioning algorithm that is external to UniVerse. The EXTERNAL clause has the following syntax: EXTERNAL <i>routine.name</i> <i>routine.name</i> The external routine used as the partitioning algorithm.
SYSTEM [<i>s</i>]	Specifies the default partitioning algorithm. <i>s</i> is the character used as a separator in record IDs. A hyphen (-) separator is the default.

Partitioning Algorithm Parameters (Continued)

Description

REBUILD.DF reads the distributed file header and opens each part file to verify its path and part number. If there are inconsistencies, REBUILD.DF fixes them. It then removes part files with duplicate part numbers from the distributed file.

Example

This example fixes PARTFILE2's part number, corrects PARTFILE4's pathname, and removes PARTFILE1 from DIST.FILE. It then determines that two part files have the same part number (3) and removes them from DIST.FILE. The [DEFINE.DF](#) command adds the part files removed by REBUILD.DF back to DIST.FILE. Notice that the ADDING clause specifies the correct part number for PARTFILE2.

```
>REBUILD.DF DIST.FILE SYSTEM -

Compiling "@PART.ALGORITHM".
IF INDEX ( @ID , - , 1 ) THEN FIELD ( @ID , - , 1 ) ELSE ERROR
Compiling "@PART.ALGORITHM".
IF INDEX ( @ID , - , 1 ) THEN FIELD ( @ID , - , 1 ) ELSE ERROR
Compiling "@PART.ALGORITHM".
IF INDEX ( @ID , - , 1 ) THEN FIELD ( @ID , - , 1 ) ELSE ERROR

Rebuilding "DIST.FILE" - First Pass.
```

```

Examining "PARTFILE2".
Part number in PARTFILE2 has changed from 2 to 3.
Fixing part number for PARTFILE2.

Examining "PARTFILE4".
Partfile "PARTFILE4" has been moved to /usr/ACCOUNTS/PARTFILE4.
Fixing pathname for "PARTFILE4".

Examining "PARTFILE1".
Part file "PARTFILE1" has an invalid part number of 0.
Invalid or missing Partblock for "PARTFILE1".
Removing PARTFILE1 from DIST.FILE.

Examining "PARTFILE3".
Rebuilding "DIST.FILE" - Second Pass.

Examining "PARTFILE2".
Warning: multiple occurrences of part number 3.
Removing PARTFILE2 from DIST.FILE.

Examining "PARTFILE4".

Examining "PARTFILE3".
Warning: multiple occurrences of part number 3.
Removing PARTFILE3 from DIST.FILE.

Updating Distributed File "DIST.FILE".
>DEFINE.DF DIST.FILE ADDING PARTFILE1 1 PARTFILE2 2 PARTFILE3 3
Compiling "@PART.ALGORITHM".
IF INDEX ( @ID , - , 1 ) THEN FIELD ( @ID , - , 1 ) ELSE ERROR
Part file "PARTFILE1", Path "/usr/SALES/PARTFILE1", Part number 1
Added.

Part file "PARTFILE2", Path "/usr/SALES/PARTFILE2", Part number 2
Added.

Part file "PARTFILE1", Path "/usr/SALES/PARTFILE3", Part number 3
Added.

>

```

RECORD

Use RECORD to verify whether a record already exists in a file. RECORD displays the group the record ID hashed to and indicates whether the record was found in the file.

Syntax

RECORD [[**DICT**] *filename* [*record*]]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
<i>filename</i>	The name of the file.
<i>record</i>	The record you want to verify.

RECORD Parameters

Description

If you omit a qualifier, RECORD prompts for it.

Example

```
>RECORD PAYABLES 11145
Record "11145" hashes to group 1 but was not found.

>RECORD
File Name = ORDERS
Record ID = 12223
Record "12223" hashes to group 4 but was not found.
>
```

RECOVER.FILE

Use the RECOVER.FILE command to recover files activated for Transaction Logging. You must be a UniVerse Administrator logged on to the UV account to use this command.

Syntax

RECOVER.FILE [-F *file_path*] [-M [*list_name* | -A]] [-C | -U *starting_log_number ending_log_number*] [-S | -R] [-I *record_ID* | -N *record_list_name*] [-L *log_path*] [-V *verbosity*] [-B *start_time*] [-E *end_time*] [-T *tape_device*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
-F <i>file_path</i>	Specifies a single file to recover. You must specify fully-qualified path, and activate the file to recover. Use a space between the -F option and <i>file_path</i> . Do not use with the -M option.
-M <i>list_name</i> -A	Specifies a <i>list_name</i> stored in the &SAVEDLISTS& file. If you do not specify a list name, you must specify -A to specify all active files. Use a space between the -M parameter and <i>list_name</i> , or the -M parameter and -A. Do not use with -F option.
-C <i>starting_log_number ending_log_number</i>	Instructs the roll forward process to begin processing at <i>starting_log_number</i> and stop at the end of <i>ending_log_number</i> . During the recovery process, UniVerse verifies the log numbers. Use a space between the -C and <i>starting_log_number</i> and <i>ending_log_number</i> . Do not use with the -U option.
-U <i>starting_log_number ending_log_number</i>	Instructs the roll forward process to begin processing at <i>starting_log_number</i> and stop at the end of <i>ending_log_number</i> . During the recovery process, UniVerse does not verify the log numbers. Use a space between the -C and <i>starting_log_number</i> and <i>ending_log_number</i> . Do not use with the -C option.

RECOVER.FILE Parameters

Parameter	Description
-S	Specifies to output messages to the screen. If you do not specify -S, UniVerse writes output messages to the uvrolf.info file. Do not use with the -R option.
-R	Specifies to retain the uvrolf.info file. UniVerse appends output messages to the existing uvrolf.info file. Do not use with the -S option.
-I <i>record_ID</i>	Specifies a single record for UniVerse to recover. The record must exist in the file you specify with the -F option. Use a space between the -I parameter and <i>record_ID</i> . If you do not specify a record ID, UniVerse displays an error and the program terminates. Do not use with the -N option.
-N <i>record_list_name</i>	Specifies a list of records stored in <i>record_list_name</i> in the &SAVEDLISTS& file to recover. UniVerse recovers the records from the file you specify with the -F option. Use a space between the -N parameter and <i>record_list_name</i> . If you do not specify <i>record_list_name</i> , UniVerse displays an error and the program terminates. Do not use with the -I option.
-L <i>log_path</i>	Specifies the directory for <i>log_path</i> . If you do not specify the -L option, UniVerse uses the default log path. Do not enter a space between -L and <i>log_path</i> .
-B <i>start_time</i>	Specifies the starting date and time within <i>starting_log_number</i> you specify with the -C or -U option from which to begin restoring the logs. Enter <i>start_time</i> in the yyyy-mm-ddThh:mm:ss or UTC format. The hour must be 00 - 23. Use a space between -B and <i>start_time</i> .

RECOVER.FILE Parameters (Continued)

Parameter	Description
-E <i>end_time</i>	Specifies the ending date and time within <i>ending_log_number</i> you specify with the -C or -U parameter to stop restoring the logs. Enter <i>end_time</i> in the yyyy-mm-ddThh:mm:ss or UTC format. The hour must be 00 - 23. Use a space between -E and <i>end_time</i> .
-V <i>level</i>	Specifies the amount of messaging to output by the rollforward process. <i>level</i> must be 0, 1, 2, or 3. 0 is the minimal message output, while 3 is the maximum message output. If you do not specify <i>level</i> , UniVerse displays an error and the process terminates. Do not enter a space between -V and <i>level</i> .
-T <i>device</i>	Specifies the logs to recover the exist on <i>device</i> . <i>device</i> must exist in the &DEVICE& file. To specify multiple tape devices, use a separate -T option followed by <i>device</i> . You must enter a space between -T and <i>device</i> . Do not use with the -L option.

RECOVER.FILE Parameters (Continued)

Description

Use the RECOVER.FILE command to start the roll forward process.

You should use the RECOVER.FILE command only when the system is not in use. Use the SUSPEND.RECOVERY command to suspend transaction logging, or SHUTDOWN.RECOVERY to stop transaction logging.

RECOVERY.CHECKPOINT

Use RECOVERY.CHECKPOINT to find the number of the first log file you need for a roll-forward recovery. You must be a UniVerse Administrator logged in to the UV account to use RECOVERY.CHECKPOINT.

Syntax

RECOVERY.CHECKPOINT { *listname* | ALL }

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>listname</i>	The name of a saved select list containing full paths of all currently activated recoverable UniVerse files that require roll-forward recovery.
ALL	Specifies that all recoverable files listed as currently activated in the UV.TRANS file are checked.

RECOVERY.CHECKPOINT Parameters

Description

RECOVERY.CHECKPOINT searches all selected UniVerse files for the file containing the earliest log file checkpoint. You should roll forward this log file first. RECOVERY.CHECKPOINT also reports the last log file checkpointed. You can then determine how many log files to restore in view of the disk space available.

RECOVERY.CHECKPOINT also lists the names of any files that are flagged as inconsistent.

Example

This example searches all recoverable files listed as currently active in the UV.TRANS file, and returns the numbers of the lowest and highest checkpoint IDs in the specified files:

```
>RECOVERY.CHECKPOINT ALL  
Low Checkpoint mark is 7  
High Checkpoint mark is 20
```

RECOVERY.CONSISTENT

Use RECOVERY.CONSISTENT to clear the flag that indicates a file is in an inconsistent state. You must be a UniVerse Administrator logged in to the UV account to use RECOVERY.CONSISTENT.

Syntax

RECOVERY.CONSISTENT *pathname*

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>pathname</i>	The full pathname of a recoverable UniVerse file flagged as inconsistent. To determine which files are flagged as inconsistent, use the RECOVERY.CHECKPOINT command.

RECOVERY.CONSISTENT Parameter

Example

This example clears the ORDERS file's inconsistency flag:

```
>RECOVERY.CONSISTENT /usr/accounts/ORDERS
```

REFORMAT

Use REFORMAT to create an inverted file. You can direct the output of the REFORMAT command to another file or to magnetic tape.

Syntax

REFORMAT [DICT | USING [DICT] *dictname*] *filename* [*records* | FROM *n*]
[*selection*] [*output.limiter*] [*sort*] *output* [*modifiers*] [MTU *mtu*] [BLK *size*]
[*labelopt*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Reformats records in the file dictionary of <i>filename</i> . If you do not specify DICT, records in the data file are reformatted.
USING [DICT] <i>dictname</i>	If DICT is not specified, uses <i>dictname</i> as the dictionary of <i>filename</i> . If DICT is specified, the dictionary of <i>dictname</i> is used as the dictionary of <i>filename</i> .
<i>filename</i>	The file whose records you want to reformat. You can specify <i>filename</i> anywhere in the sentence. REFORMAT uses the first word in the sentence that has a file descriptor in the VOC file as the file name.
<i>records</i>	Specifies the records to reformat. You can specify as many record IDs as you want, separated by spaces. Enclose record IDs in single quotation marks to ensure that Retrieve does not interpret them as field names or keywords.
FROM <i>n</i>	Specifies the records whose record IDs are stored in select list <i>n</i> .

REFORMAT Parameters

Parameter	Description
<i>selection</i>	A selection expression specifying the conditions that data in a record must meet for the record to be selected. A selection expression begins with the keyword WITH. For syntax details, see the WITH keyword in Chapter 2, “ UniVerse Keywords .”
<i>output.limiter</i>	An expression specifying the conditions that values in multivalued fields must meet for those values to be output. An output-limiting expression begins with the keyword WHEN. For syntax details, see the WHEN keyword in Chapter 2, “ UniVerse Keywords .”

REFORMAT Parameters (Continued)

Parameter	Description								
<i>sort</i>	A sort expression that specifies the type of sort and the field to sort on. The syntax is as follows: <table><tr><td>BY <i>field</i></td><td>Sort on <i>field</i> in ascending order.</td></tr><tr><td>BY.DSND <i>field</i></td><td>Sort on <i>field</i> in descending order.</td></tr><tr><td>BY.EXP <i>field</i></td><td>Explode multivalues in <i>field</i> and <i>sort</i> in ascending order.</td></tr><tr><td>BY.EXP.DSND <i>field</i></td><td>Explode multivalues in <i>field</i> and <i>sort</i> in descending order.</td></tr></table> You can use REFORMAT with the BY.EXP and BY.EXP.DSND keywords to reformat data in multivalued fields into singlevalued fields by using an exploded sort. For more information about sort expressions, see Chapter 2, “UniVerse Keywords.” When NLS locales are enabled, REFORMAT uses the Collate convention of the current locale to determine the collating order for any existing sort expression. If an index has its own Collate convention defined, REFORMAT uses it instead of the current locale definition. For more information, see the <i>UniVerse NLS Guide</i>.	BY <i>field</i>	Sort on <i>field</i> in ascending order.	BY.DSND <i>field</i>	Sort on <i>field</i> in descending order.	BY.EXP <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in ascending order.	BY.EXP.DSND <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in descending order.
BY <i>field</i>	Sort on <i>field</i> in ascending order.								
BY.DSND <i>field</i>	Sort on <i>field</i> in descending order.								
BY.EXP <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in ascending order.								
BY.EXP.DSND <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in descending order.								
<i>output</i>	Specifies the fields whose data you want to include. You must specify at least one field. The fields must be defined in the file dictionary. If you are listing dictionary records, the fields must be defined in the DICT.DICT file. You can precede a field name with the CALC field modifier. You cannot follow a field name with field qualifiers.								
<i>modifiers</i>	One or more of the following keywords: <table><tr><td>FIRST</td><td>ID.SUP</td><td>SAMPLE</td></tr><tr><td>ID.ONLY</td><td>ONLY</td><td>SAMPLED</td></tr></table> These keywords modify the report format. For information about them and their synonyms, see Chapter 2, “UniVerse Keywords.”	FIRST	ID.SUP	SAMPLE	ID.ONLY	ONLY	SAMPLED		
FIRST	ID.SUP	SAMPLE							
ID.ONLY	ONLY	SAMPLED							
REFORMAT Parameters (Continued)									

Parameter	Description
MTU <i>mtu</i>	Specifies a drive other than 0. <i>mtu</i> indicates the mode (<i>m</i>), number of tracks (<i>t</i>), and tape unit number (<i>u</i>) of a tape drive. You can use the default for these values, 000, which indicates ASCII mode, 9-track tape, and unit number 0, otherwise REFORMAT ignores the track specification. See the MTU keyword in Chapter 2, “UniVerse Keywords.”
BLK <i>size</i>	Specifies the block size if it is different from the default of 8192 bytes.
<i>labelopt</i>	One of the following: PICK.FORMAT (for Pick tape format) INFORMATION.FORMAT (for Prime INFORMATION tape format) REALITY.FORMAT (for REALITY tape format)

REFORMAT Parameters (Continued)

Description

After you enter a REFORMAT command, the following prompt appears:

File Name =

Enter a valid UniVerse file name as a destination file, or enter the keyword TAPE (or BDE in IN2 flavor accounts). REFORMAT directs output of the command either to the file or to magnetic tape, respectively. If you direct the output to another file, you must specify a file name other than the source file name. If you press ENTER at the File Name = prompt, no processing occurs.

When you send output to another file, REFORMAT uses the first field defined by the output specification as the record ID in the new file. The other output fields become fields in the new file. The order of fields in the output specification of the REFORMAT command determines the order of fields in the new file. The new file must already exist, and you must create new dictionary fields manually if you need them.

When you send output to magnetic tape, REFORMAT writes one tape record for each reformatted record. Use the [T.ATT](#) command or the [BLK](#) keyword to specify tape record size. The default tape record size is 8192 bytes. The new record ID and the fields for the data record are concatenated, ending with a segment mark. They are either padded or truncated, depending on the size of the tape record. You can use [MTU](#), [BLK](#), and *labelopt* when directing output to tape.

REFORMAT ignores control breaks and totals.

RELEASE

Use RELEASE to unlock records locked by a BASIC MATREADL statement, MATREADU statement, READL statement, READU statement, READVL statement, and READVU statement. You must use RELEASE from the same terminal you were using when the MATREADL, MATREADU, READL, READU, READVL, and READVU statements locked the records.

Syntax

RELEASE [*filename*] [*record*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>filename</i>	The name of the file from which to release the locks.
<i>record</i>	The record ID that you want to release the locks from.

RELEASE Parameters

Description

You should not need to use RELEASE unless an unusual event occurs that prevents the BASIC program from releasing records locked by MATREADL, MATREADU, READL, READU, READVL, and READVU statements.

To release all the locks that you have set, enter **RELEASE**.

To release all the locks that you have set for a specific file, enter **RELEASE *filename***.

To release a lock you have set for a specific record in a specific file, enter **RELEASE *filename record***.

You can examine a list of the records that have been locked using [LIST.READU](#).

RELEASE.LFILE

Use RELEASE.LFILE to release a transaction logging log file for reuse. You must be a UniVerse Administrator logged in to the UV account to use RELEASE.LFILE.

Syntax

RELEASE.LFILE *logfile*

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>logfile</i>	The number of the log file you want to release for reuse. The log file's status must be Full.

RELEASE.LFILE Parameter

Description

RELEASE.LFILE makes a full log file available for reuse. Once you release a log file, its logged updates are lost; if you still need them, you should back up the log file to tape before releasing them.



Note: *You cannot use RELEASE.LFILE to release the tape log file.*

Executing RELEASE.LFILE for *logfile* releases files of the form *logdir/lgnnn*, where *logdir* is the path of the current log directory and *nnn* is a log file number.

RELEASE.LFILE reflects the changed state of the log files it releases by updating the records corresponding to these log files in the UV_LOGS file. RELEASE.LFILE must therefore have access to UV_LOGS. The STATUS field of these corresponding records in UV_LOGS are updated by RELEASE.LFILE. The status of the released log files changes from Full to Released, and new Available log files are created to replace the Released log files, which no longer exist physically on disk.

If the state of transaction logging is full, executing `RELEASE.LFILE` after a Full log file has been backed up to tape does not change the systemwide state of transaction logging. After releasing Full log files, use [ENABLE.RECOVERY](#) or the Enable logging option on the Manage Logging Activity menu to start up the log daemon.

If `RELEASE.LFILE` cannot release a specified log file, execution stops with an error.

Example

This example releases log file 351 for reuse:

```
>RELEASE.LFILE 351
```

REMOVE.DEMO.FILES

Use REMOVE.DEMO.FILES to delete the 10 sample files for the Circus database from the account you are logged in to.

Syntax

REMOVE.DEMO.FILES

Description

REMOVE.DEMO.FILES removes the UniVerse data files, and the associated file dictionaries that were created by the [MAKE.DEMO.FILES](#) command.

REMOVE.DEMO.TABLES

Use REMOVE.DEMO.TABLES to delete the 10 sample tables for the Circus database from the account you are logged in to. You must be the owner of the tables or have DBA privilege to use REMOVE.DEMO.TABLES.

Syntax

REMOVE.DEMO.TABLES

Description

REMOVE.DEMO.TABLES removes the tables that were created by the [MAKE.DEMO.TABLES](#) command.

REPEAT

Use REPEAT in a loop. See the [LOOP](#) command for information about REPEAT.

RESIZE

Use RESIZE to reorganize a file with a new file type, modulo, and separation, or to change an existing nondynamic file to a dynamic file. You cannot use RESIZE to change the parameters of a dynamic file, or to resize an SQL table to a type 1, type 19, or type 25 file.

Syntax

RESIZE [DICT] [*filename*] [*type*] [*modulo*] [*separation*]
[CONCURRENT | INPLACE | USING *partition*] [64BIT | 32BIT]

RESIZE [DICT] [*filename*] [30 | DYNAMIC] [*parameter* [*value*]] ...
[USING *partition*] [64BIT | 32BIT]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Resizes the file dictionary.
<i>filename</i>	The name of the file to be resized.
<i>type</i>	The new file type, a number (1 through 19, or 25) specifying one of the nondynamic file types.
<i>modulo</i>	The new modulo, an integer from 1 through 8,388,608 defining the number of groups in the file. <i>modulo</i> is ignored if you specify a nonhashed or dynamic file type.
<i>separation</i>	The new separation, an integer from 1 through 8,388,608 specifying the group buffer size in 512-byte blocks. <i>separation</i> is ignored if you specify a nonhashed or dynamic file type.
CONCURRENT	Specifies that other users can use the file while it is being resized. This option does not exist when resizing to a type 30 file. This option is not supported on Windows platforms.

RESIZE Parameters

Parameter	Description
INPLACE	Resizes the file in its present physical location. This option does not exist when resizing to a type 30 file.
USING <i>partition</i>	Specifies the path of the work area that RESIZE will use for creating the necessary temporary files. For example, the following command changes SUN.MEMBER to a dynamic file, and creates the temporary files it needs in the partition <i>/u4</i> : >RESIZE SUN.MEMBER DYNAMIC USING /u4 RESIZE moves the files back into the correct directory after resizing the SUN.MEMBER file and deleting the original SUN.MEMBER file.
30	Makes the file a dynamic file.
DYNAMIC	Makes the file a dynamic file.

RESIZE Parameters (Continued)

Specify the following parameters only for dynamic files:

Parameter	Description
GENERAL	Specifies the general hashing algorithm for a dynamic file. GENERAL is the default.
SEQ.NUM	Specifies a hashing algorithm suitable for sequential numbers for a dynamic file. Use this hashing algorithm only for records with IDs that are mainly numeric, sequential, and consecutive.
GROUP.SIZE { 1 2 }	Specifies the size of each group in the file, either 1 or 2. 1 specifies a group size of 2048 bytes, which is equivalent to a separation of 4. 2 specifies a group size of 4096 bytes, which is equivalent to a separation of 8. A group size of 2048 (GROUP.SIZE 1) is the default.
MINIMUM.MODULUS <i>n</i>	Specifies the minimum modulo of the file, an integer value greater than 1. This value is also the initial value of the modulo of the dynamic file. A minimum modulo of 1 is the default.

RESIZE Parameters for Dynamic Files

Parameter	Description
SPLIT.LOAD <i>n</i>	Specifies the level at which the file's modulo is increased by 1. SPLIT.LOAD takes a numeric argument indicating the percentage of space allocated for the file. When the data in the file exceeds the specified percentage of the space allocated for the file, the data in one of the groups is divided equally between itself and a new group, to increase the modulo by 1. The default SPLIT.LOAD is 80%.
MERGE.LOAD <i>n</i>	Specifies the level at which the file's modulo is decreased by 1. MERGE.LOAD takes a numeric argument indicating the percentage of space allocated for the file. When the data in the file is less than the specified percentage of the space allocated for the file, the data in the last group of the file is merged with another group, to decrease the modulo by 1. The default MERGE.LOAD is 50%.

RESIZE Parameters for Dynamic Files (Continued)

Parameter	Description
LARGE.RECORD <i>n</i>	Specifies the size of a record considered too large to be included in the primary group buffer, specified as an integer or a percentage. Specified as an integer, the value is the number of bytes a record must contain to be considered a large record. Specified as a percentage, the value is a percentage of the group size. When the size of a record exceeds the specified value, the data for the record is put in an overflow buffer, but the record ID is put in the primary buffer. This method of large record storage increases access speed. The default LARGE.RECORD size is 80%.
RECORD.SIZE <i>n</i>	Calculates the values for group size and large record size based on the value of the estimated average record size specified. The value is your estimate of the average record size for the dynamic file, specified in bytes. RECORD.SIZE does not limit the size of records. If you specify a value for group size (GROUP.SIZE) or for large record size (LARGE.RECORD), those values override the value calculated by RECORD.SIZE.
MINIMIZE.SPACE	Calculates the best amount of space required by the file (at the expense of access time), using the values for the split load, merge load, and large record size. If you specify values for split load, merge load, or large record size, those values override the value calculated by MINIMIZE.SPACE. If you specify MINIMIZE.SPACE and RECORD.SIZE, the value for large record size calculated by MINIMIZE.SPACE is used above the value calculated by RECORD.SIZE.

RESIZE Parameters for Dynamic Files (Continued)

The following options override the current setting of the 64BIT_FILES configurable parameter:

Parameter	Description
64BIT	Creates a 64-bit file on a UniVerse system using 32-bit file systems.
32BIT	Creates a 32-bit file on a UniVerse system using 64-bit file systems.

64BIT_FILES Configurable Parameter

Description

As you add records to a file, your original estimate of the file type, modulo, and separation may no longer be correct. You can use commands such as [HASH.HELP](#), [HASH.HELP.DETAIL](#), [HASH.TEST](#), and [HASH.TEST.DETAIL](#) to test a file type, modulo, and separation, without actually changing the file structure. As a file grows, it may be necessary to change the modulo, separation, and file type to maintain the most efficient file structure.

Dynamic files make file management easier for users. The default parameters are correct for most dynamic files. You can increase the efficiency of some files by setting parameters different from the defaults. Use the [ANALYZE.FILE](#) command to verify that your settings are actually more efficient than the defaults.

Test several combinations of file type, modulo, and separation using [HASH.TEST](#) or [HASH.TEST.DETAIL](#) before using [RESIZE](#).

To use [RESIZE](#) on a specific file, specify file name, file type, modulo, and separation qualifiers. To retain an existing file type, modulo, or separation, enter * as its value. If you are reorganizing a data file, do not include the keyword [DICT](#).

You need not copy the file or reallocate records when you use [RESIZE](#). [RESIZE](#) distributes all the records from the original file groups into new groups.

[RESIZE](#) normally copies records from one disk location to another. The [CONCURRENT](#) option lets other users access the file while it is being resized. If many users are accessing the file and many records are locked, performance will be notably slower.

The [INPLACE](#) option uses a space minimization algorithm and resizes the file in its current physical location. This is useful when the amount of available disk space is low and there is not enough room for large temporary files.

[RESIZE](#) changes the physical file structure and deletes any physical disk records that have become superfluous by the reorganization of a group.

RESTORE.LOCALE

Use RESTORE.LOCALE in NLS mode to restore a previously saved locale setting.

Syntax

RESTORE.LOCALE

Description

RESTORE.LOCALE restores a locale that was previously saved using the [SAVE.LOCALE](#) command. When you enter UniVerse, a locale is set up and saved automatically. RESTORE.LOCALE restores this initial locale unless you executed a SAVE.LOCALE command during your UniVerse session.

NLS Mode

RESTORE.LOCALE returns an error if the NLSLCMODE or NLSMODE configurable parameter in the *uvconfig* file is set to 0. For more information about locales, see the *UniVerse NLS Guide*.

REVISE

Use REVISE to add, change, and delete file records.

Syntax

```
REVISE [ DICT ] [ filename ] [ fields ] [ USING process ]  
[ verifield { VERIFY | VERIFILE } verifile [ VERIFIELD display.field ] ]
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies that you want to revise the file dictionary.
filename	The name of the file to revise.
fields	The names of fields to revise. If you do not specify a field, ReVise uses the fields defined in the @REVISE phrase in the file dictionary. ReVise use the files defined in the @ phrase in the file dictionary. If the dictionary contains neither an @REVISE phrase nor an @ phrase, ReVise creates an @REVISE phrase containing all the data descriptor fields in the dictionary and uses these fields.
process	The name of a process definition for prompting. The process must exist in the file REVISE.PROCESSES. A process serves as a template that ReVise uses to prompt for entries. You can enter any of the following proccesses: ENTER.DICT Creates an entry in a file dictionary. ENTER.PDICT Creates an entry in a Pick-style file dictionary. ENTER.DICTAS Creates A-type and S-type dictionary entries in a UniVerse file dictionary. ENTER.MENUS Creates a menu in a menu file. ENTER.FMENUS Creates a formatted menu in a menu file.

REVISE Parameters

Parameter	Description
	ENTER.S Creates a stored sentence in the VOC file.
	ENTER.SELECTOR Creates a menu selector record in the VOC file.
VERIFY	or (VERIFILE) Verifies that data you enter exists as a record ID in a verification file.
<i>verifield</i>	Either the name of a field in the current file whose data is being validated, or the name of the record ID field (@ID) in the current file.
	<i>field</i> If you use a field name, ReVise verifies that the value entered in <i>verifield</i> is also a record ID in <i>verifile</i> .
	<i>record.ID</i> If you use the record ID field name, ReVise verifies that the entered value is <i>not</i> a record ID in <i>verifile</i> . This does not work, however, if the ReVise sentence also includes a VERIFIELD expression. In such a case, <i>record.ID</i> must be a record ID in <i>verifile</i> .
<i>verifile</i>	The name of the verification file whose record IDs ReVise uses to validate the data entered in <i>verifield</i> .
VERIFIELD	Displays the values in verifile that verify <i>verifield</i> data.
<i>display:field</i>	The name of the field in <i>verifile</i> whose data ReVise displays for each value entered in <i>verifield</i> .

REVISE Parameters (Continued)

If you do not specify *filename* or the keyword USING, ReVise displays the following prompt:

Enter process name:

Enter one of the processes from the file REVISE.PROCESSES.

REVOKE.ENCRIPTION.KEY

Use the REVOKE.ENCRIPTION.KEY command to revoke access to the encryption key from other users. When a key is created, only the owner of the key has access. The owner of the key can revoke access from other users.

Syntax

REVOKE.ENCRIPTION.KEY {PUBLIC | *grantee* {*grantee...*}}

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
PUBLIC	Revokes PUBLIC access to the encryption key from all users on the system. For example, if “PUBLIC” access is granted, it is removed. However, this does not revoke individual user or group access that had been granted.
<i>grantee</i>	Revokes access to the encryption key from the <i>grantee</i> you specify. <i>grantee</i> can be a user name, or, on UNIX platforms, a group name. If you specify a group name, prefix the name with an asterisk (“*”). When you specify a group name, UniVerse revokes access from all users belonging to the group. On Windows platforms, a group name can be a local group or a global group (specified in the form of *Domain\global-group). A user can also be a domain user, specified in the form of Domain\user. In the case of “\” appearing in a group or user name, you should use quotation marks to enclose the name. Grantees cannot revoke access to the encryption key from other users.

REVOKE.ENCRIPTION.KEY Parameters

RUN

Use RUN to execute a compiled BASIC program.

Syntax

RUN [*filename*] *program* [*options*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>filename</i>	The name of the file containing the source program. RUN appends .O to <i>filename</i> , then executes the specified program from this object file. If you do not specify <i>filename</i> , RUN assumes the BP file.
<i>program</i>	The name of the record containing the source code of the program. You must specify <i>program</i> .

RUN Parameters

options can be any of the following:

Option	Description
NO.WARN	Suppresses all warning (nonfatal) error messages. If you do not specify NO.WARN, run-time error messages are printed on the screen as they are encountered.
NO.PAGE	Suppresses automatic paging. Programs that position the cursor with @ functions need not disable paging.

RUN Options

Option	Description
LPTR	Spools program output to the printer rather than to the terminal screen.
KEEP.COMMON	Retains the value of variables in unnamed common if a CHAIN statement passes program execution to another BASIC program.
TRAP	Enters the RAID debugger instead of displaying warning (nonfatal) error messages.
RUN Options (Continued)	

Description

If you do not use NO.PAGE, UniVerse counts the lines printed by the PRINT statement and displays the following message at the bottom of each full page:

Press any key to continue...

Some programs use the cursor position function, @. These programs do not need the NO.PAGE option. UniVerse counts the lines printed by PRINT statements only for programs without any cursor control.

SAVE.LIST

Use SAVE.LIST to save an active select list. You can use [GET.LIST](#) to recall this list for subsequent processing and avoid having to repeat the selection process each time you want to use the same select list.

Syntax

```
SAVE.LIST [ [filename ] listname ] [ FROM n ]
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>filename</i>	The name of an existing type 1 or type 19 file. The default file is &SAVEDLISTS& .
<i>listname</i>	The name you are giving to the list. If you do not give the list a name, SAVE.LIST assigns the name &TEMP <i>port#</i> &. The <i>port#</i> is your terminal number.
FROM <i>n</i>	Specifies select list <i>n</i> . If you do not specify a FROM clause, select list 0 is saved.

SAVE.LIST Parameters

Description

To save select list 0, you must use SAVE.LIST immediately after you have created the list.

SAVE.LIST stores the list as a record in a type 1 or type 19 file.

If no select list is active, SAVE.LIST deletes a saved list in the &SAVEDLISTS& file with the same name as *listname*. For more information, see [&SAVEDLISTS&](#) in Chapter 4, “[Local and System Files](#).”

Example

```
>SELECT SUN.MEMBER WITH YR.JOIN > 1980
```

```
9 record(s) selected to SELECT list #0.
```

```
>>SAVE.LIST YR.80
```

```
9 record(s) SAVED to SELECT list "YR.80".
```

SAVE.LOCALE

Use SAVE.LOCALE in NLS mode to save the current locale setting. You can use [RESTORE.LOCALE](#) to restore the saved locale to the previous state.

Syntax

SAVE.LOCALE

Description

SAVE.LOCALE saves the current setting for a locale. You can save only one locale at a time.

For more information about locales, see the *UniVerse NLS Guide*.

SAVE.STACK

Use SAVE.STACK to save the current sentence stack in the [&SAVEDLISTS&](#) file, under a specified name.

Syntax

SAVE.STACK [*listname*]

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>listname</i>	The name of the saved sentence stack.

SAVE.STACK Parameter

Description

If you do not specify *listname*, SAVE.STACK prompts you for one.

Example

```
>SAVE.STACK
Name to save command stack as: MGR.STACK
Command stack "MGR.STACK" saved in file "&SAVEDLISTS&".
>
```

SEARCH

Use SEARCH to create a select list of records that contain an occurrence of a specified string. This command is the same as a [SELECT](#) command with a WITH or WHEN *field* MATCHING clause except that SEARCH checks every field of each record for a match and is quicker than using the MATCHING operator.

Syntax

```
SEARCH [ DICT | USING [ DICT ] dictname ] filename [ records | FROM n ]  
[ selection ] [ output.limiter ] [ sort ] [ TO n ] [ options ]
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Searches records in the file dictionary of <i>filename</i> . If you do not specify DICT, records in the data file are searched.
USING [DICT] <i>dictname</i>	If DICT is not specified, uses <i>dictname</i> as the dictionary of <i>filename</i> . If DICT is specified, the dictionary of <i>dictname</i> is used as the dictionary of <i>filename</i> .
<i>filename</i>	The file whose records you want searched. You can specify <i>filename</i> anywhere in the sentence. SEARCH uses the first word in the sentence that has a file descriptor in the VOC file as the file name.
<i>records</i>	Specifies the records to include in the select list. You can specify as many record IDs as you want, separated by spaces. Enclose record IDs in single quotation marks to ensure that RetriVe does not interpret them as field names or keywords.
FROM <i>n</i>	Specifies the records whose record IDs are stored in select list <i>n</i> .

SEARCH Parameters

Parameter	Description								
<i>selection</i>	A selection expression specifying the conditions that data in a record must meet for the record to be selected. A selection expression begins with the keyword WITH. For syntax details, see the WITH keyword in Chapter 2, “ UniVerse Keywords .”								
<i>output.limiter</i>	An expression specifying the conditions that values in multivalued fields must meet for those values to be output. An output-limiting expression begins with the keyword WHEN. For syntax details, see the WHEN keyword in Chapter 2, “ UniVerse Keywords .”								
<i>sort</i>	A sort expression that specifies the type of sort and the field to sort on. The syntax is as follows: <table> <tr> <td>BY <i>field</i></td><td>Sort on <i>field</i> in ascending order.</td></tr> <tr> <td>BY.DSND <i>field</i></td><td>Sort on <i>field</i> in descending order.</td></tr> <tr> <td>BY.EXP <i>field</i></td><td>Explode multivalues in <i>field</i> and <i>sort</i> in ascending order.</td></tr> <tr> <td>BY.EXP.DSND <i>field</i></td><td>Explode multivalues in <i>field</i> and <i>sort</i> in descending order.</td></tr> </table> For more information about sort expressions, see Chapter 2, “UniVerse Keywords.” When NLS locales are enabled, SEARCH uses the Collate convention of the current locale to determine the collating order for any existing sort expression. If an index has its own Collate convention defined, SEARCH uses it instead of the current locale definition. For more information, see the <i>UniVerse NLS Guide</i>.	BY <i>field</i>	Sort on <i>field</i> in ascending order.	BY.DSND <i>field</i>	Sort on <i>field</i> in descending order.	BY.EXP <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in ascending order.	BY.EXP.DSND <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in descending order.
BY <i>field</i>	Sort on <i>field</i> in ascending order.								
BY.DSND <i>field</i>	Sort on <i>field</i> in descending order.								
BY.EXP <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in ascending order.								
BY.EXP.DSND <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in descending order.								
TO <i>n</i>	The number to assign to the list, 0 through 10. If you do not specify a number, SEARCH creates select list 0.								
<i>options</i>	One or more of the following: <table> <tr> <td>ALL.MATCH</td><td>Selects only records containing all of the specified strings.</td></tr> <tr> <td>NO.MATCH</td><td>Selects only records not containing any of the specified strings.</td></tr> </table>	ALL.MATCH	Selects only records containing all of the specified strings.	NO.MATCH	Selects only records not containing any of the specified strings.				
ALL.MATCH	Selects only records containing all of the specified strings.								
NO.MATCH	Selects only records not containing any of the specified strings.								

SEARCH Parameters (Continued)

Parameter	Description
NO.SELECT	Lists record IDs instead of creating a select list.
EXPLODE	Makes each multivalue that matches a specified string an element in the select list. The select list elements also include the field and value number in which the string was found. When you use EXPLODE, SEARCH creates the list elements according to the following format: <i>record.id</i>v<i>value</i>s<i>field</i>v0 The <i>record.id</i> is the record ID where the string is found, <i>value</i> and <i>field</i> are the value and field numbers within the record that it is found in, v represents a value mark, and s represents a subvalue mark.
SQUAWK	Lists a message for each record selected.

SEARCH Parameters (Continued)

Description

After you enter **SEARCH**, the following prompt appears:

STRING :

You can enter as many separate strings as you want. Press ENTER at the prompt to terminate input. SEARCH selects only records containing any of the values you entered for string.

ESearch is a synonym for the SEARCH command.

SELECT

Use SELECT to create a list of records that meet specified criteria. You can then use this list with other commands in the UniVerse system. Because the list contains the data or record IDs from selected records, it is called a select list.

Syntax

```
SELECT [ DICT | USING [ DICT ] dictname ] filename [ records | FROM n ]  
[ selection ] [ output.limiter ] [ sort ] [ SAVING [ UNIQUE ] field ]  
[ NO.NULLS ] ] [ TO n ] [ report.qualifiers ]
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Selects records in the file dictionary of <i>filename</i> . If you do not specify DICT, records in the data file are selected.
USING [DICT] <i>dictname</i>	If DICT is not specified, uses <i>dictname</i> as the dictionary of <i>filename</i> . If DICT is specified, the dictionary of <i>dictname</i> is used as the dictionary of <i>filename</i> .
<i>filename</i>	The file whose records you want to select. You can specify <i>filename</i> anywhere in the sentence. SELECT uses the first word in the sentence that has a file descriptor in the VOC file as the file name.
<i>records</i>	Specifies the records to include in the list. You can specify as many record IDs as you want, separated by spaces. Enclose record IDs in single quotation marks to ensure that Retrieve does not interpret them as field names or keywords.
FROM <i>n</i>	Specifies the records whose record IDs are stored in select list <i>n</i> .
<i>selection</i>	A selection expression specifying the conditions that data in a record must meet for the record to be selected. A selection expression begins with the keyword WITH. For syntax details, see the WITH keyword in Chapter 2, “UniVerse Keywords.”

SELECT Parameters

Parameter	Description								
<i>output.limiter</i>	An expression specifying the conditions that values in multivalued fields must meet for those values to be output. An output-limiting expression begins with the keyword WHEN. For syntax details, see the WHEN keyword in Chapter 2, “ UniVerse Keywords .”								
<i>sort</i>	A sort expression that specifies the type of sort and the field on which field to sort. The syntax is as follows: <table> <tr> <td>BY <i>field</i></td><td>Sort on <i>field</i> in ascending order.</td></tr> <tr> <td>BY.DSND <i>field</i></td><td>Sort on <i>field</i> in descending order.</td></tr> <tr> <td>BY.EXP <i>field</i></td><td>Explode multivalues in <i>field</i> and <i>sort</i> in ascending order.</td></tr> <tr> <td>BY.EXP.DSND <i>field</i></td><td>Explode multivalues in <i>field</i> and <i>sort</i> in descending order.</td></tr> </table> For more information about sort expressions, see Chapter 2, “UniVerse Keywords.” When NLS locales are enabled, SELECT uses the Collate convention of the current locale to determine the collating order for any existing sort expression. If an index has its own Collate convention defined, SELECT uses it instead of the current locale definition. For more information, see the <i>UniVerse NLS Guide</i>.	BY <i>field</i>	Sort on <i>field</i> in ascending order.	BY.DSND <i>field</i>	Sort on <i>field</i> in descending order.	BY.EXP <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in ascending order.	BY.EXP.DSND <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in descending order.
BY <i>field</i>	Sort on <i>field</i> in ascending order.								
BY.DSND <i>field</i>	Sort on <i>field</i> in descending order.								
BY.EXP <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in ascending order.								
BY.EXP.DSND <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in descending order.								
SAVING	Specifies that the list be made up of field values instead of record IDs. Do not use a SAVING clause on multivalued fields.								
UNIQUE	Omits duplicates of the saved field from the select list.								
<i>field</i>	The name of a field whose values you want to constitute the select list.								
NO.NULLS	Omits empty string values in the saved field from the select list.								
TO <i>n</i>	The number to assign to the list, 0 through 10. If you do not specify a number, SELECT creates select list 0.								
<i>report.qualifiers</i>	One or more of the following keywords: FIRST LPTR SAMPLE SAMPLED These keywords modify the report format. For information about them, see Chapter 2, “UniVerse Keywords.”								

SELECT Parameters (Continued)

SEMAPHORE.STATUS

Use SEMAPHORE.STATUS to display the status of system semaphores. You can execute SEMAPHORE.STATUS only from the UV account.

Syntax

SEMAPHORE.STATUS

Description

SEMAPHORE.STATUS is not available on all systems. It is designed only for systems whose semaphores are implemented in assembly language.

SEMAPHORE.STATUS lists information for each numbered semaphore, and for the login, port status, type 30 file and transaction logging semaphores. The SEMAPHORE.STATUS report displays this information under these column headings:

Column Heading	Description
State	Nonzero for locked, zero for unlocked.
Netnode	Identifies the host from which the lock originated. Zero indicates a lock on the local machine. The network node number is the last part of the TCP/IP host number specified in the <i>/etc/hosts</i> file.
Owner	The user ID of the user who locked the semaphore.
Collisions	The number of times two processes collide when trying to set a lock.
Retrys	The number of times processes repeatedly try to set a lock.

SEMAPHORE.STATUS Information

Example

The following is an example of a SEMAPHORE.STATUS report:

>SEMAPHORE.STATUS

File access		State	Netnode	Owner	Collisions	Retrys
Semaphore #	1	0	0	0	0	0
Semaphore #	2	0	5	0	0	0
Semaphore #	3	0	0	0	0	0
Semaphore #	4	0	0	0	0	0
Semaphore #	5	0	0	0	0	0
Semaphore #	6	0	0	0	0	0
Semaphore #	7	0	0	0	0	0

.
.
.

Group access		State	Netnode	Owner	Collisions	Retrys
Semaphore #	1	0	0	0	4	10
Semaphore #	2	0	0	0	6	6
Semaphore #	3	0	0	0	210	330
Semaphore #	4	0	0	0	10	13
Semaphore #	5	0	0	0	7	7
Semaphore #	6	0	0	0	4	10

.
.
.

Login		State	Netnode	Owner	Collisions	Retrys
Semaphore #	1	0	0	0	0	0

Port status		State	Netnode	Owner	Collisions	Retrys
Semaphore #	1	0	0	0	0	0

Type 30 file		State	Netnode	Owner	Collisions	Retrys
Semaphore #	1	0	0	0	0	0

Transaction log		State	Netnode	Owner	Collisions	Retrys
Semaphore #	1	0	0	0	0	0

SET.FILE

Use SET.FILE to create a Q-pointer in the VOC file. Q-pointers are file definition synonyms that point to files in local and remote UniVerse accounts.

Syntax

SET.FILE [*account*] [*filename*] [*pointer*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>account</i>	The name of the account containing the remote file, as defined in the UV.ACCOUNT file; the path where an account resides; or the name of a login account.
<i>filename</i>	The name of a file in <i>account</i> .
<i>pointer</i>	The record ID of the Q-pointer in your VOC file.

SET.FILE Parameters

Description

If you do not specify qualifiers, SET.FILE prompts for them.

SET.FILE.MAP

Use SET.FILE.MAP in NLS mode to assign a map to a file.

Syntax

SET.FILE.MAP [**DICT**] *filename* { *mapname* | **DEFAULT** | **NONE** }
[**FORCE**] [**VERIFY**]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
<i>filename</i>	The name or path of the file to which you want to assign a map.
<i>mapname</i>	The name of a map that is built and installed in shared memory. <i>mapname</i> can be the value UNICODE or UTF8 for type 1 or type 19 files only.
DEFAULT	Replaces the current map with the map named by the corresponding configurable parameter from the <i>uvconfig</i> file.
NONE	Specifies no map for a file. Records are written back to the file in internal format.
FORCE	Assigns a new map to a file. You must use this option to assign the new map if the file already has a map (including NONE) explicitly assigned to it.
VERIFY	Checks that the map is valid for the data in the file. If the new map would result in lost data, the map is not changed unless you also specify the FORCE option.

SET.FILE.MAP Parameters

Description

Use SET.FILE.MAP to assign a map to a file. The map defines the external character set for the file. Use this command when you have existing UniVerse files containing non-ASCII characters and you want to access them in NLS mode.

Files with no assigned map use default maps defined by configurable parameters in the *uvconfig* file. When you create new files, CREATE.FILE also uses these default maps. The default maps defined in the *uvconfig* file are as follows:

- For existing hashed files: NLSDEFFILEMAP
- For existing type 1 or type 19 files: NLSDEFDIRMAP
- For new hashed files: NLSNEWFILEMAP
- For new type 1 or type 19 files: NLSNEWDIRMAP

If any record IDs cannot be mapped using the map you specify, SET.FILE.MAP does not change the map. Instead, you see a message similar to the following indicating which record IDs are affected:

```
Unmappable Record ID found "ÃÃÃ"(C0C1C2)
```

The quoted characters are the bytes of the record ID as stored in the file. Whether you can read this or not depends on your terminal and its map. The string in parentheses is the record ID's byte value in hexadecimal format.

You can remove a map from a file using either the NONE or DEFAULT option.

If a file has secondary indexes, the indexes must either have the same map as the file, or have mapping set to NONE. You must then use BUILD.INDEX to rebuild the secondary indexes for NLS mode.

SET.FILE.MAP does not convert a file's record IDs or data. To do that, use the UNICODE.FILE command.

@SYSTEM.RETURN.CODE returns 0 if the command succeeds, or a value less than 0 if there is an error.

For more information about maps, see the *UniVerse NLS Guide*.

Examples

This example associates the KSC5601 map, a Korean multibyte character set, with the ACCOUNTS file. When the file is read, all existing records in the file are mapped from the KSC5601 character set to internal format. When the file is written, records are converted back to KSC5601 using the same map.

```
>SET.FILE.MAP ACCOUNTS KSC5601
```

The next example uses NONE to specify that no mapping is to take place for the ACCOUNTS file. You must use the FORCE keyword, as the ACCOUNTS file already has a map associated with it.

```
>SET.FILE.MAP ACCOUNTS NONE FORCE
```

If the ACCOUNTS file was populated between the first and second example, you would need to use the UNICODE.FILE command instead. This ensures that data is converted to internal form.

The next example removes the map name from the file. The file now uses the map specified in the NLSDEFFILEMAP parameter in the *uvconfig* file.

```
>SET.FILE.MAP ACCOUNTS DEFAULT
```

SET.GCI.MAP

Use SET.GCI.MAP in NLS mode to set a global map for all GCI operations or to display the GCI map setting.

Syntax

SET.GCI.MAP [*mapname* | DEFAULT | NONE | OS]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>mapname</i>	The name of the map you want to set.
DEFAULT	Sets the GCI map to the default specified by the NLSDEFGCIMAP configurable parameter in the <i>uvconfig</i> file.
NONE	Sets the GCI map to raw mode using the internal character set.
OS	Sets the GCI map to the operating system map specified by the NLS-DEFOSMAP configurable parameter in the <i>uvconfig</i> file.

SET.GCI.MAP Parameters

Description

Use SET.GCI.MAP to set a map for GCI subroutines to use for passing strings between BASIC and C or FORTRAN programs.

Use SET.GCI.MAP with no options to display the current GCI map setting as specified in the *uvconfig* file or changed by a subsequent SET.GCI.MAP command.

Note: *You cannot associate a specific map with a specific GCI routine.*

@SYSTEM.RETURN.CODE returns 0 if the command succeeds, or a value less than 0 if there is an error.



For information about GCI, see the *UniVerse GCI Guide*. For more information about NLS, see the *UniVerse NLS Guide*.

SET.INDEX

Use SET.INDEX to change the characteristics of all the secondary indexes in a file.

Syntax

SET.INDEX [**DICT**] *filename* [**TO** [*pathname* | **NULL**]] [*options*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
<i>filename</i>	The name of a UniVerse file in which to change the characteristics of the secondary indexes. If a select list for <i>filename</i> is active, you do not need to specify <i>filename</i> .
<i>pathname</i>	The full path for the directory of the secondary indexes in <i>filename</i> to be changed in the file header.
NULL	Removes the current <i>pathname</i> for the indexes in <i>filename</i> from the file header.

SET.INDEX Parameters

options can be any of the following:

Option	Description
FORCE	Specifies that settings should be changed to <i>pathname</i> or NULL without prompting. If you do not specify FORCE , SET.INDEX verifies that you want to change or remove the path.

SET.INDEX Options

Option	Description
INFORM	Displays the current <i>pathname</i> for the indexes in <i>filename</i> in the file header on the terminal. You cannot use INFORM with another option.
OFF	Disables automatic secondary index updates for <i>filename</i> . Same as DISABLE.INDEX.
ON	Enables automatic secondary index updates for <i>filename</i> . Same as ENABLE.INDEX.

SET.INDEX Options (Continued)

Description

You can use SET.INDEX on any UniVerse file for which you have write permissions. If you move or copy a file at the operating system level, SET.INDEX lets you easily change the file header information to point to the new location for the secondary indexes.

Examples

This example displays the current pathname for the indexes in the INVENTORY file on the terminal:

```
>SET.INDEX INVENTORY INFORM
Indices for file 'INVENTORY' reside in
'/usr/ardent/uv/I_INVENTORY'.
```

The following sequence removes the path information for the indexes in the INVENTORY file from the file header. The INFORM option displays the change.

```
>SET.INDEX INVENTORY TO NULL

The current indices for file 'INVENTORY' are at path:

/usr/ibm/uv/I_INVENTORY

Do you wish to remove this path (Y/N)? Y

File header block updated.

>SET.INDEX INVENTORY INFORM

File INVENTORY has no secondary indices.
```

The following example uses the FORCE keyword to remove the path information without prompting:

```
>SET.INDEX INVENTORY TO NULL FORCE
File header block updated.
```

The next example changes the path of the indexes for the INVENTORY file in the file header and verifies that you want to make the change:

```
>SET.INDEX INVENTORY TO /u1/ibm/uv/I_INVENTORY

The current indices for file 'INVENTORY' are at path
  No indices path currently defined.
to be changed to path
  /usr/ibm/uv/uv/I_INVENTORY
Do you wish to make this change (Y/N)? Y

File header block updated.
```

The following sequence enables automatic updating of the indexes of the INVENTORY file. The LIST.INDEX command displays the update mode of the indexes.

```
>SET.INDEX INVENTORY ON
Automatic secondary index updates for file INVENTORY have been
enabled.
>LIST.INDEX INVENTORY ALL
Alternate Key Index Summary for file INVENTORY
File..... INVENTORY
Indices..... 1 (0 A-type, 0 C-type, 1 D-type, 0 I-type, 0 S-
type)
Index Updates.. Enabled, No updates pending
Index name      Type  Build    Nulls  In DICT  S/M  Just Unique
Field num/I-type
F1              D    Required Yes    No      S    L    N    1
```

SET.LOCALE

Use SET.LOCALE in NLS mode to disable a locale or set a new locale.

Syntax

SET.LOCALE [*category* | ALL] { *locale* | OFF }

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>category</i>	Sets the locale to one of the following categories: COLLATE CTYPE MONETARY NUMERIC TIME
ALL	Sets all categories. ALL is the default.
<i>locale</i>	The name of the locale you want to set.
OFF	Disables the locale for the designated category.

SET.LOCALE Parameters

Description

When you want to specify numeric and monetary formatting for a locale, you must set both the Numeric and Monetary categories to something other than OFF, for example, DEFAULT. If you do not, UniVerse treats BASIC conversions such as MD, ML, and MR as if locales are turned off.

For complete information about locales, see the *UniVerse NLS Guide*.

Examples

This example sets the current locale to FR-FRENCH in all categories:

```
>SET.LOCALE ALL FR-FRENCH
```

The next example disables locales for all categories until you execute another SET.LOCALE, or you restore a previously saved locale that has categories set:

```
>SET.LOCALE ALL OFF
```

The next example sets the locale for the Time category to FR-FRENCH:

```
>SET.LOCALE TIME FR-FRENCH
```

SET.LOG.ATTR

Use SET.LOG.ATTR to set or change the transaction logging operating mode. You must be a UniVerse Administrator logged in to the UV account to use this command.

Syntax

SET.LOG.ATTR { CHECKPOINT | ARCHIVE } { ON | OFF }
[DEVICELIST *devices*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
CHECKPOINT	Changes checkpoint mode. You must set checkpoint mode to ON to enable warmstart transaction logging and recovery.
ARCHIVE	Changes archive mode. Archive mode corresponds to Release 7 transaction logging. Use archive mode for recovery from media failure.
<i>devices</i>	Specifies a list of tape devices, separated by spaces. Each device must be defined in the &DEVICE& file. Transaction updates are logged to the tape devices in the order in which you specify them. ARCHIVE must be set to ON, otherwise the DEVICELIST clause is ignored. When you specify <i>devices</i> , CHECKPOINT mode is automatically set to OFF.

SET.LOG.ATTR Parameters

Description

The transaction logging system can run in checkpoint, archive mode, or both. You can change the mode of transaction logging only when its state is inactive, uninitialized, suspended, or disabled.

Examples

This example sets both transaction logging modes to ON. File updates are logged to the log file on disk:

```
>SET.LOG.ATTR CHECKPOINT ON
Checkpoint mode has been set to ON.
>SET.LOG.ATTR ARCHIVE ON
Archive mode has been set to ON.
```

The next example sets archive mode to ON and defines MT0 and MT1 as tape devices. File updates are logged directly to tape, starting with device MT0.

```
>SET.LOG.ATTR ARCHIVE ON DEVICELIST MT0 MT1
Checkpoint mode has been set to OFF.
Archive mode has been set to ON with logging to tape device(s):
  1) MT0
  2) MT1
```

SET.REMOTE.ID

Use SET.REMOTE.ID to define the user name and password to use for access to files on a remote UniVerse system. SET.REMOTE.ID also sets a user’s SQL user name on the remote system.

Syntax

```
SET.REMOTE.ID [ domain ] username { password | PROMPT } ON nodename  
[ [ GROUP ] groupname ]
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>domain</i>	Windows Platforms. The domain name. When connecting to Windows systems, you can precede the user name with the domain name. If you do not specify a domain name, the domain name defaults to that of the local system.
<i>username</i>	The user name to use for access to files on <i>nodename</i> . This must be a valid user name on the remote system. Windows Platforms. When you are connecting to a Windows system, <i>username</i> is not case-sensitive.
<i>password</i>	The password associated with <i>username</i> to use for access to files on <i>nodename</i> . <i>password</i> is always case-sensitive.
PROMPT	Prompts the user to enter a valid password. The password is not echoed to the screen.
ON	Specifies the name of a remote system running UniVerse. The remote system must be defined on the local system.
GROUP	Limits <i>username</i> ’s group membership on a remote UNIX system to <i>groupname</i> . <i>groupname</i> must be a valid group on the UNIX system, and <i>username</i> must be a member of the group.

SET.REMOTE.ID Parameters

Description

Use SET.REMOTE.ID before entering any command that accesses a remote file. If you are connecting a local UNIX or Windows system to a remote Windows system via TCP/IP, you must use SET.REMOTE.ID to specify a user name and password.

Once you are connected to a remote system, you cannot use SET.REMOTE.ID to change the remote user name, password, or group name.

If *username*, *password*, *nodename*, or *groupname* does not match the appropriate definitions on the remote or local system, all access to files on the remote system are denied.

If you do not specify *groupname*, the user belongs to all groups to which *username* is assigned.

You can include one or more SET.REMOTE.ID commands in a LOGIN paragraph if you want to access remote systems during every UniVerse session. You can also use the UVNETRID environment variable to permanently set effective user names. See *UV/Net II Guide* for more information.

Example

This example sets the user name of the current user to *susan* on the remote machine called *sales*:

```
>SET.REMOTE.ID susan xyz4569a ON sales
```

SET.SEQ.MAP

Use SET.SEQ.MAP in NLS mode to assign maps to files or devices opened with the BASIC OPENSEQ statement or OPENDEV statement.

Syntax

SET.SEQ.MAP [*mapname* | DEFAULT | NONE]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>mapname</i>	The name of a map that is built and installed in shared memory.
DEFAULT	Uses the map name specified by the NLSDEFSEQMAP configurable parameter in the <i>uvconfig</i> file.
NONE	Uses internal format when reading or writing the file.

SET.SEQ.MAP Parameters

Description

SET.SEQ.MAP specifies the map you want to use with the BASIC sequential I/O statements if no explicit map is assigned to the sequential file or device that you opened. Use SET.SEQ.MAP with no options to report the current map name as set by the *uvconfig* parameter or by a previous SET.SEQ.MAP command.

On UNIX systems, you can use SET.SEQ.MAP to assign maps to pipes opened with the OPENSEQ or OPENDEV statement.

Note: *Sequential I/O is used mainly to communicate with processes outside UniVerse. It is unlikely that such processes will understand UniVerse internal format. If you want to communicate in true UTF8, use that as mapname. To use Unicode wide characters, specify UNICODE as mapname.*



@SYSTEM.RETURN.CODE returns 0 if the command succeeds, or a value less than 0 if there is an error.

For more information about maps, see the *UniVerse NLS Guide*.

Example

This example sets the map to KSC5601. Any subsequent BASIC OPENSEQ *pathname* statement now uses map KSC5601 instead of that specified by the NLSDEFSEQMAP parameter.

```
>SET.SEQ.MAP KSC5601
```

SET.SQL

Use SET.SQL to set SQL environment variables for the current UniVerse session.

Syntax

SET.SQL { *options* }

Option	Description
INF {ON OFF}	Turns first-normal-form mode on or off. This setting overrides the setting specified by the SQLSetConnectOption function. The default is OFF.
EMPTY.NULL {ON OFF}	Turns empty-null mapping on or off. This setting overrides the setting specified by the SQLSetConnectOption function. The default is OFF.
ISOLATION <i>level</i>	Sets the isolation level for the current session. <i>level</i> is a number from 0 through 4. 0 NO ISOLATIO 1 READ UNCOMMITTED 2 READ COMMITTED 3 REPEATABLE READ 4 SERIALIZABLE The default value is 0. This setting overrides the setting specified by the ISOMODE configurable parameter.
JOIN.BUFFER <i>bytes</i>	Specifies the size of the in-memory join buffer, which is the amount of locally cached data from joined tables that can be stored before the data starts using disk storage. It cannot be larger than 32,384. The default value is 4095. This setting overrides the setting specified by the JOINBUF configurable parameter.

SET.SQL Options

Option	Description
LOCK.WAIT <i>seconds</i>	Specifies the number of seconds to wait on a record or file lock before returning an error. The default values is 3600.
OPTIM.SCAN {ON OFF}	Turns optimistic scanning on or off. The default is ON.
REPORTING	Lists the current SQL settings.
SELECT.BUFFER <i>bytes</i>	Specifies the size of the in-memory select list buffer, which is the amount of locally cached select data that can be stored before the select list starts using disk storage. The default value is 4095. This setting overrides the setting specified by the SELBUF configurable parameter.
VOC.CACHE {ON OFF}	Turns caching of VOC file verbs and keywords on or off. When this option turns caching on, it also flushes the cache. The default is ON.

SET.SQL Options (Continued)

Description

SET.SQL lets you set the SQL environment variables and other aspects of the SQL environment dynamically at run time.

The VOC cache is flushed whenever the VOC.CACHE option is set to ON, and whenever the VOC file is updated.

Optimistic scanning scans a table or file, assuming it will not run into a record lock. If it does not find a lock, data is returned as usual. If it does find a lock, the block is rescanned after the lock goes away or until the timeout specified by the LOCK.WAIT option.

Note: ODBC client applications can use the CALL statement to run the SET.SQL command with the following options only:

- LOCK.WAIT
- OPTIM.SCAN
- SELECT.BUFFER
- JOIN.BUFFER



■ VOC.CACHE

ODBC client applications must not use SET.SQL to set the ISOLATION, 1NF, or EMPTY.NULL options.

Examples

This example sets the isolation level to 3 and turns first-normal-form mode and empty-null mapping on:

```
>SET.SQL ISOLATION 3 1NF ON EMPTY.NULL ON
UniVerse/SQL: SQL env: ISOLATION "3", 1NF ON, EMPTY.NULL ON,
                        LOCK.WAIT 3600 seconds, JOIN.BUFFER 4095 bytes,
                        OPTIM.SCAN ON, SELECT.BUFFER 4096 bytes.
```

The next example doubles the size of the select list and join buffers:

```
>SET.SQL SELECT.BUFFER 8190 JOIN.BUFFER 8190
UniVerse/SQL: SQL env: ISOLATION "0", 1NF OFF, EMPTY.NULL OFF,
                        LOCK.WAIT 3600 seconds, JOIN.BUFFER 8190 bytes,
                        OPTIM.SCAN ON, SELECT.BUFFER 8190 bytes.
```

The next example lists the current SET.SQL settings:

```
>SET.SQL
UniVerse/SQL: SQL env: ISOLATION "0", 1NF OFF, EMPTY.NULL OFF,
                        LOCK.WAIT 3600 seconds, JOIN.BUFFER 4095 bytes,
                        OPTIM.SCAN ON, SELECT.BUFFER 4096 bytes.
```

SET.TERM.TYPE

Use SET.TERM.TYPE to specify your terminal type. When you set your terminal type, UniVerse sets terminal characteristics appropriate for the terminal you are using.

Syntax

SET.TERM.TYPE [*code*] [*options*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description												
<i>code</i>	The code for the type of terminal. It is case-sensitive. If you do not specify <i>code</i> , the current terminal type is assumed.												
<i>options</i>	One or more of the following: <table><tr><td>AUTONL</td><td>Generates a new line sequence if a line longer than the width of the screen is typed.</td></tr><tr><td>AUXMAP <i>mapname</i></td><td>In NLS mode, sets a map for an auxiliary printer attached to the terminal. <i>mapname</i> is the name of the map you want to set. If you do not specify <i>mapname</i>, the map for the terminal is used.</td></tr><tr><td>FUNDAMENTAL</td><td>Loads the default key bindings.</td></tr><tr><td>HUSH</td><td>Suppresses terminal output.</td></tr><tr><td>LENGTH <i>n</i></td><td>Sets the length of the terminal screen to <i>n</i> lines.</td></tr><tr><td>MAP <i>mapname</i></td><td>In NLS mode, sets a map for the terminal. <i>mapname</i> is the name of the map you want to set.</td></tr></table>	AUTONL	Generates a new line sequence if a line longer than the width of the screen is typed.	AUXMAP <i>mapname</i>	In NLS mode, sets a map for an auxiliary printer attached to the terminal. <i>mapname</i> is the name of the map you want to set. If you do not specify <i>mapname</i> , the map for the terminal is used.	FUNDAMENTAL	Loads the default key bindings.	HUSH	Suppresses terminal output.	LENGTH <i>n</i>	Sets the length of the terminal screen to <i>n</i> lines.	MAP <i>mapname</i>	In NLS mode, sets a map for the terminal. <i>mapname</i> is the name of the map you want to set.
AUTONL	Generates a new line sequence if a line longer than the width of the screen is typed.												
AUXMAP <i>mapname</i>	In NLS mode, sets a map for an auxiliary printer attached to the terminal. <i>mapname</i> is the name of the map you want to set. If you do not specify <i>mapname</i> , the map for the terminal is used.												
FUNDAMENTAL	Loads the default key bindings.												
HUSH	Suppresses terminal output.												
LENGTH <i>n</i>	Sets the length of the terminal screen to <i>n</i> lines.												
MAP <i>mapname</i>	In NLS mode, sets a map for the terminal. <i>mapname</i> is the name of the map you want to set.												

SET.TERM.TYPE Parameters

Parameter	Description
NEEDNL	Lets UniVerse generate new line sequences.
VERIFY.SUP	Specifies no prompting is required if an invalid terminal type is entered.
WIDTH <i>n</i>	Sets the width of the terminal screen to <i>n</i> characters. <i>n</i> must be an integer between 11 and 32,767.

SET.TERM.TYPE Parameters (Continued)

Description

If you enter SET.TERM.TYPE without a code, UniVerse prompts for the code. Enter **?** at the prompt to list supported codes.

The terminal code is the name of the file in the *terminfo* directory that contains the terminal definition.

If NLS is enabled, SET.TERM.TYPE overrides the *terminfo* entry for *mapname*. If you omit *code* and specify MAP or AUXMAP, the existing terminal type is used. *mapname* must have been built and loaded into shared memory. Use the value DEFAULT if you want to use the map for the corresponding terminal type in the *terminfo* directory. If there is no default map specified in *terminfo*, SET.TERM.TYPE uses the default specified by the NLSDEFTERMMAP parameter in the *uvconfig* file.

For more information about terminfo records that you can use to set maps for terminals and auxiliary printers, see the *UniVerse NLS Guide*.

Examples

This example sets a terminal type to VT100:

```
>SET.TERM.TYPE VT100
```

This example sets a terminal type to VT220 and sets up an auxiliary printer map. The terminal map is set up from the *terminfo* record or from the NLSDEFTERMMAP parameter in the *uvconfig* file.

```
>SET.TERM.TYPE VT220 AUXMAP JIS-EUC
```

SETFILE

Use SETFILE to create a file pointer entry in the VOC file. A file pointer entry is a synonym for a file in your own or in another account. SETFILE creates a record in the VOC file that points to a UniVerse file that already exists.

Syntax

SETFILE [*pathname*] [*filename*] [OVERWRITING]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>pathname</i>	The full path of the data file for which you are creating an entry.
<i>filename</i>	The name that you will use to access the file from this account.
OVERWRITING	Replaces an existing entry having the same name as <i>filename</i> with the new entry. Before using OVERWRITING, examine the VOC file and be sure it is okay to overwrite the VOC file. SETFILE does not verify what it is overwriting.

SETFILE Parameters

Example

```
>SETFILE
```

```
Enter DATA file pathname = /usr/a/ACCOUNTS
What do you want to call it in your VOC file = NEW.ACCT
Establishing file pointer:
Pathname      "/usr/a/ACCOUNTS"
VOC Name      "NEW.ACCT"
OK to establish pointer (Y/N) ?N
SETFILE aborted.
>
```

SETPTR (UNIX)

Use SETPTR to set the line printer spooler options for a logical print channel. These changes are effective until you use SETPTR again or use LOGOUT or QUIT.

Syntax

SETPTR [*channel* ,*l.len* ,*p.len* ,*top* ,*bottom* ,*mode* ,*options*]

Parameters

All arguments are positional parameters. Each parameter is optional, but its position must be held by a comma. If you enter SETPTR with no parameters, the current settings for logical print channel 0 are displayed.

Parameter	Description
<i>channel</i>	Identifies the logical print channel assigned to the printer with the ASSIGN command. Enter a number from 0 through 255. The default is 0. If you specify channel with no options, SETPTR displays the current settings for that logical print channel.
<i>l.len</i>	Set the line length, that is, the paper width. The default is 132.
<i>p.len</i>	Sets the number of lines per page. The default is 66.

SETPTR Parameters

Parameter	Description
<i>top</i>	Sets the top margin in number of lines. The default is 3.
<i>bottom</i>	Sets the bottom margin in number of lines. The default is 3.
mode	A number from 1 through 5, used to direct output to one of the following places. 1) Line Printer Spooler Output (default). 2) Assigned Device. To send output to an assigned device, you must first assign the device to a logical print channel, using the ASSIGN command. The ASSIGN command does an automatic SETPTR command using the default parameters, except for mode, which it sets to 2. Use SETPTR only if you have to change the default parameters. 3) Hold File Output. Mode 3 directs all printer output to a file called &HOLD&. If a &HOLD& file does not exist in your account, SETPTR creates the file and its dictionary (D_&HOLD&). You must execute SETPTR with mode 3 before each report to create unique report names in &HOLD&. If a report exists with the same name, the new report overwrites it. 4) Synonym for mode 2. 5) Synonym for mode 2.

SETPTR Parameters (Continued)

options can be any of the following:

Option	Description
AS [<i>name</i>]	Same as BANNER.
AT <i>name</i>	Routes output to <i>name</i> of a printer on a Windows system, using the format map name associated with the entry in the &DEVICE& file.
BANNER [<i>name</i>]	In mode 1, name appears on the second line of the banner page under the account name, that is, the login name. In mode 3, specifies the record ID of the record in &HOLD& which stores the report. If you do not specify <i>name</i>, the record ID is P#0000. If you specify <i>name</i>, it is the record ID of the output record. In either case, each subsequent print job uses the same record ID and overwrites the previous job.

SETPTR Options

Option	Description
BANNER NEXT [<i>name</i>]	In mode 3, appends a sequential number to the name under which successive reports are created in &HOLD&. If you do not specify <i>name</i> , the record ID is P#0000_ <i>nnnnnn</i> , where <i>nnnnnn</i> is increased for each new print job. If you specify <i>name</i> , the record ID is <i>name_</i> <i>nnnnnn</i> .
BANNER UNIQUE [<i>name</i>]	In mode 3, append a sequential number to the name under which successive records are created in &HOLD&. If you do not specify <i>name</i> , the record ID is P#0000_ <i>nnnnnn</i> , where <i>nnnnnn</i> is increased by each subsequent SETPTR command. If you specify <i>name</i> , the record ID is <i>name_</i> <i>nnnnnn</i> .
BRIEF	Suppresses the display of SETPTR settings.
COPIES <i>n</i>	Specifies the number of copies to print (with only one banner page).
DEFER <i>time</i>	Defers printing until <i>time</i> . Specify <i>time</i> in one of the following formats. The formats beginning with a plus sign (+) specify time relative to the current system time. <i>hh:mm</i> <i>dd.hh:mm</i> <i>mm.dd.hh:mm</i> <i>yy.mm.dd.hh:mm</i> <i>dd</i> <i>mm:dd</i> <i>yy.mm.dd</i> + <i>mm</i> + <i>hh:mm</i> + <i>dd.hh:mm</i>
EJECT	Ejects a page at the end of the print job.
ENDPAGE	Specifies the last page to print.
FMT	Specifies that the spooler controls pagination and formatting instead of the application.
FORM <i>name</i>	Uses the form <i>name</i> for printing this job. FORM prompts the user to put the named form in the printer.

SETPTR Options (Continued)

Option	Description
FORMAT.MAP <i>mapname</i>	Sets a map name for formatting only. Data still goes to the spool queue with a map name of NONE.
FORMFEED	Specifies that formfeeds be added to printer output.
FTN	Specifies that the print job contains FORTRAN control codes.
HEADN	Same as HEADON.
HEADON	Turns banner printing back on if NOHEAD or NHEAD is set.
HOLD	In mode 1, sends print jobs to the spooler as hold files. The spooler does not print held jobs when they are sent. You can use the PRINT.ADMIN command to print held jobs. After a held job is printed, it is removed from the spool queue.
INFORM	In mode 1, displays the spooler job number of newly queued jobs.
KEEP	Keeps the print file open. This option lets you append subsequent reports to the same print file.
LNUM	Prints line numbers.
NFMT	Specifies that the application controls pagination and formatting instead of the spooler.
NHEAD	Suppress printing a banner.
NODEFAULT	Changes only those parameters specified with your command, and does not supply default settings for parameters not specified.
NOEJECT	Does not eject a page at the end of the print job.
NOFMT	Same as NFMT.
NOFORMFEED	Suppresses the addition of formfeeds to printer output.
NOHEAD	Same as NHEAD.
NOHOLD	In mode 1, turns off the HOLD option.

SETPTR Options (Continued)

Option	Description
NOKEEP	Closes an open print file.
NORETAIN	In mode 1, turns off the RETAIN option.
PRINTER <i>name</i>	Same as AT.
PRIORITY	Sets the print job's priority. The highest priority is 1, and the lowest is 255.
REQUEUE	Same as RETAIN.
RETAIN	In mode 1, sends jobs to the spooler as a retained hold file. The spooler does not print held jobs when they are sent. You can use the PRINT.ADMIN command to print held jobs. After a retained held job is printed, it remains in the spool queue.
STARTPAGE	Specifies the number of the page at which to begin printing.
USEROPTS <i>options</i>	A string of one or more options to be sent to a printer driver script. <i>options</i> can be up to 128 characters long. If you specify more than one option, enclose <i>options</i> in double quotation marks.

SETPTR Options (Continued)

You can send the following SETPTR options as shell arguments to printer driver scripts:

Argument	SETPTR Options
\$7	Line length
\$8	Page length
\$9	Eject flag (1 = EJECT, 0 = NOEJECT)
Shift the argument stack down to reference the following two arguments:	
\$1	Banner flag (1 = print banner, 0 = suppress banner)
\$2	USEROPTS options

SETPTR Options for Printer Driver Scripts

See *Administering UniVerse* for how to use SETPTR options and other spooler information in printer driver scripts.

NLS Mode

When NLS is enabled, *mode* lets you associate a map with a print channel. You can determine display widths for formatting spooled output. The internal to external mapping takes place when a report prints. *mode* can be any of the modes described in the following table:

Mode	Description
1	Mode 1 specifies that a map is assigned for formatting only. Data goes to the spool queue with a map of NONE. FORMAT.MAP sets a map name for formatting only. Data still goes to the spool queue with a map of NONE. If you omit FORMAT.MAP, the format map is set by the AT <i>name</i> option from the printer map in the &DEVICE& file—that is, the map name associated with the printer name in the &DEVICE& file. Otherwise the format map is set from the default NLSDEFPTRMAP parameter in the <i>uvconfig</i> file.
2	Mode 2 sets a format map from the assigned device, using the device entry in the &DEVICE& file.
3	Mode 3 sets a format map from the &HOLD& file. The format map is used to format the character width for the output display. This information is stored until it is sent to the printer. If a name of NONE is used, the format map name is set from <i>mapname</i> for FORMAT.MAP, otherwise the format map is set from the default NLSDEFPTRMAP parameter in the <i>uvconfig</i> file.

NLS Modes

For more information about character mapping, see the *UniVerse NLS Guide*.

Examples

```
>SETPTR 0,132,66,3,3,1
Unit Number      : 0
Page Width       : 132
Page Depth       : 66
Top Margin       : 3
Bottom Margin    : 3
Print mode       : 1 - Spooled Output

Default spool banner : "uniVerse"
OK to set parameters as displayed? Y
>SETPTR 1,21,12,3,3,1,BRIEF
```

SETPTR (Windows Platforms)

Use SETPTR to set the line printer spooler options for a logical print channel. These changes are effective until you use SETPTR again or use LOGOUT or QUIT.

Syntax

SETPTR [*channel* ,*l.len* ,*p.len* ,*top* ,*bottom* ,*mode* ,*options*]

Parameters

All arguments are positional parameters. Each parameter is optional, but its position must be held by a comma. If you enter SETPTR with no parameters, the current settings for logical print channel 0 are displayed.

Parameter	Description
<i>channel</i>	Identifies the logical print channel assigned to the printer with the ASSIGN command. Enter a number from 0 through 255. The default is 0. If you specify <i>channel</i> with no options, SETPTR displays the current settings for that logical print channel.
<i>l.len</i>	Sets the line length, that is, the paper width. The default is 132.
<i>p.len</i>	Sets the number of lines per page. The default is 66.

SETPTR Parameters

Parameter	Description
<i>top</i>	Sets the top margin in number of lines. The default is 3.
<i>bottom</i>	Sets the bottom margin in number of lines. The default is 3.
<i>mode</i>	A number from 1 through 5 that is used to direct output to one of the following places: 1) Line Printer Spooler Output (default). 2) Assigned Device. To send output to an assigned device, you must first assign the device to a logical print channel, using the ASSIGN command. The ASSIGN command does an automatic SETPTR command using the default parameters, except for mode, which it sets to 2. Use SETPTR only if you have to change the default parameters. 3) Hold File Output. Mode 3 directs all printer output to a file called &HOLD&. If a &HOLD& file does not exist in your account, SETPTR creates the file and its dictionary (D_&HOLD&). You must execute SETPTR with mode 3 before each report to create unique report names in &HOLD&. If a report exists with the same name, the new report overwrites it. 4) Synonym for mode 2. 5) Synonym for mode 2.

SETPTR Parameters (Continued)

For more information about character mapping, see the *UniVerse NLS Guide*.

options can be any of the following:

Option	Description
AS [<i>name</i>]	Same as BANNER.
AT <i>name</i>	Routes output to <i>name</i> of a printer on a Windows system, using the format map name associated with the entry in the &DEVICE& file.

SETPTR Options

Option	Description
BANNER [<i>name</i>]	In mode 1, a banner page is not produced unless the Windows printer has been configured to do so. If you do not specify <i>name</i>, the Windows printer banner is produced. If you specify <i>name</i>, it appears on the second line of the banner page, after the user name. The banner name appears as the job name in the Windows Print Manager. In mode 3, specifies the record ID of the record in &HOLD& which stores the report. If you do not specify <i>name</i>, the record ID is P#0000. If you specify <i>name</i>, it is the record ID of the output record. In either case, each subsequent print job uses the same record ID and overwrites the previous job.
BANNER NEXT [<i>name</i>]	In mode 3, appends a sequential number to the name under which successive reports are created in &HOLD&. If you do not specify <i>name</i>, the record ID is P#0000_ <i>nnnnnn</i>, where <i>nnnnnn</i> is increased for each new print job. If you specify <i>name</i>, the record ID is <i>name_</i><i>nnnnnn</i>.
BANNER UNIQUE [<i>name</i>]	In mode 3, appends a sequential number to the name under which successive records are created in &HOLD&. If you do not specify <i>name</i>, the record ID is P#0000_ <i>nnnnnn</i>, where <i>nnnnnn</i> is increased by each subsequent SETPTR command. If you specify <i>name</i>, the record ID is <i>name_</i><i>nnnnnn</i>.
BRIEF	Suppresses the display of SETPTR settings.
COPIES <i>n</i>	Specifies the number of copies to print (with only one banner page). In GDI mode, the number of copies is passed to the printer driver. In raw mode, this number is passed to the Windows system as a parameter for the print processor.

SETPTR Options (Continued)

Option	Description
DEFER <i>time</i>	Defers printing until <i>time</i>. Specify <i>time</i> in one of the following formats. The formats beginning with a plus sign (+) specify time relative to the current system time. <i>hh:mm</i> <i>dd.hh:mm</i> <i>mm.dd.hh:mm</i> <i>yy.mm.dd.hh:mm</i> <i>dd</i> <i>mm.dd</i> <i>yy.mm.dd</i> <i>+mm</i> <i>+hh:mm</i> <i>+dd.hh:mm</i> Printing can be deferred only until later the same day for Windows systems.
EJECT	Ejects a page at the end of the print job.
ENDPAGE	Specifies the last page to print.
FMT	Specifies that the spooler controls pagination and formatting instead of the application.
FONTBOLD	Uses boldface print type for printers in GDI mode.
FONTITALIC	Uses italic print type for printers in GDI mode.
FONTNAME <i>fontname</i>	Specifies the font to be used for printing in GDI mode. If you do not specify <i>fontname</i> , the printer default font is used.
FONTSIZE <i>points</i>	Used in conjunction with the <i>fontname</i> . Specifies the size of font (in points) for GDI mode. If <i>points</i> is not specified, a default size of 10 points is used.
FORM <i>name</i>	Uses the form <i>name</i> for printing this job. <i>name</i> is used as the printer name, unless a printer has been explicitly specified. If a form name and a printer name are specified, the form name is passed to the Windows printer driver.
FORMAT.MAP <i>mapname</i>	Sets a map name for formatting only. Data still goes to the spool queue with a map of NONE.

SETPTR Options (Continued)

Option	Description
FORMFEED	Specifies that formfeeds be added to printer output.
FTN	Specifies that the print job contains FORTRAN control codes.
GDI	Requests GDI printing mode for a Windows printer.
FHEADN	Same as HEADON.
HEADON	Turns banner printing back on if NOHEAD or NHEAD is set.
HOLD	In mode 1, pauses a print job. The paused print job can be resumed by using the SPOOL MODIFY or SP.EDIT command, or by resuming the job from the Windows Print Manager.
INFORM	In mode 1, displays the spooler job number of newly queued jobs.
KEEP	Keeps the print file open. You can append subsequent reports to the same print file.
LINESPACE <i>lines</i>	Specifies the line spacing in GDI mode. The line spacing is a real number giving the ratio between the desired line spacing and the default line spacing for the selected font. The default value is 1.0.
LNUM	Prints line numbers.
NFMT	Specifies that the application controls pagination and formatting instead of UniVerse.
NHEAD	Suppress printing a banner.
NODEFAULT	Changes only those parameters specified with your command, and does not supply default settings for parameters not specified. Default settings can be changed using the SETPTR.DEFAULT command.
NOEJECT	Does not eject a page at the end of the print job. This option is not supported for printers in GDI mode, and in raw mode it is available only for printers that support it.
NOFMT	Same as NFMT.

SETPTR Options (Continued)

Option	Description
NOFORMFEED	Suppresses the addition of formfeeds to printer output.
NOHEAD	Same as NHEAD.
NOHOLD	In mode 1, turns off the HOLD option.
NOKEEP	Closes an open print file.
NORETAIN	In mode 1, turns off the RETAIN option.
PRINTER <i>name</i>	Same as AT.

SETPTR Options (Continued)

Option	Description																																																		
PRIORITY	Sets the print job's priority. The highest priority is 1 and the lowest is 255. The priority you enter is converted to a number within the range for the Windows system, which goes from 99 through 1. Because the range of priorities in UniVerse goes from 1 through 254, the mapping is not one to one. The following list shows the conversion of UniVerse priorities to Windows NT priorities. <table> <tr> <th>Priority in UniVerse</th><th>Priority in Windows</th></tr> <tr><td>1</td><td>99</td></tr> <tr><td>2</td><td>98</td></tr> <tr><td>.</td><td>.</td></tr> <tr><td>.</td><td>.</td></tr> <tr><td>.</td><td>.</td></tr> <tr><td>10</td><td>90</td></tr> <tr><td>11</td><td>89</td></tr> <tr><td>12</td><td>89</td></tr> <tr><td>13</td><td>89</td></tr> <tr><td>14</td><td>88</td></tr> <tr><td>15</td><td>88</td></tr> <tr><td>16</td><td>88</td></tr> <tr><td>17</td><td>87</td></tr> <tr><td>.</td><td>.</td></tr> <tr><td>.</td><td>.</td></tr> <tr><td>.</td><td>.</td></tr> <tr><td>244</td><td>11</td></tr> <tr><td>245</td><td>10</td></tr> <tr><td>246</td><td>9</td></tr> <tr><td>247</td><td>8</td></tr> <tr><td>.</td><td>.</td></tr> <tr><td>.</td><td>.</td></tr> <tr><td>.</td><td>.</td></tr> <tr><td>254</td><td>1</td></tr> </table>	Priority in UniVerse	Priority in Windows	1	99	2	98	.	.	.	.	.	.	10	90	11	89	12	89	13	89	14	88	15	88	16	88	17	87	.	.	.	.	.	.	244	11	245	10	246	9	247	8	.	.	.	.	.	.	254	1
Priority in UniVerse	Priority in Windows																																																		
1	99																																																		
2	98																																																		
.	.																																																		
.	.																																																		
.	.																																																		
10	90																																																		
11	89																																																		
12	89																																																		
13	89																																																		
14	88																																																		
15	88																																																		
16	88																																																		
17	87																																																		
.	.																																																		
.	.																																																		
.	.																																																		
244	11																																																		
245	10																																																		
246	9																																																		
247	8																																																		
.	.																																																		
.	.																																																		
.	.																																																		
254	1																																																		
RAW	Requests raw printing mode.																																																		
REQUEUE	Same as RETAIN.																																																		

SETPTR Options (Continued)

Option	Description
RETAIN	In mode 1, same as HOLD.
STARTPAGE	Specifies the page number at which to begin printing.
TABSIZE	Specifies the spacing of tab stops in GDI mode. The default value is 8.

SETPTR Options (Continued)

When NLS is enabled, *mode* lets you associate a map with a print channel. You can determine display widths for formatting spooled output. The internal to external mapping takes place when a report prints. *mode* can be any of the modes described in the following table:

Mode	Description
1	Mode 1 specifies that a map is assigned for formatting only. Data goes to the spool queue with a map of NONE. FORMAT.MAP sets a map name for formatting only. Data still goes to the spool queue with a map of NONE. If you omit FORMAT.MAP, the format map is set by the AT <i>name</i> option from the printer map in the &DEVICE& file, that is the map name associated with the printer name in the &DEVICE& file. Otherwise, the format map is set from the default NLSDEFPTRMAP parameter in the <i>uvconfig</i> file.
2	Mode 2 sets a format map from the assigned device, using the device entry in the &DEVICE& file.
3	Mode 3 sets a format map from the &HOLD& file. The format map is used to format the character width for the output display. This information is stored until it is sent to the printer. If a name of NONE is used, the format map name is set from <i>mapname</i> for FORMAT.MAP. Otherwise, the format map is set from the default NLSDEFPTRMAP parameter in the <i>uvconfig</i> file.

NLS Modes

Examples

```
>SETPTR 0,132,66,3,3,1
Unit Number      : 0
Page Width       : 132
Page Depth       : 66
Top Margin       : 3
Bottom Margin    : 3
Print mode       : 1 - Spooled Output

Default spool banner : "uniVerse"
OK to set parameters as displayed? Y
>SETPTR 1,21,12,3,3,1,BRIEF
```

SETPTR.DEFAULT

Use SETPTR.DEFAULT to assign default SETPTR parameters over the entire system. You must be a UniVerse Administrator logged in to the UV account to use SETPTR.DEFAULT.

Syntax

SETPTR.DEFAULT

Description

SETPTR.DEFAULT uses the SETPTR settings of logical print channel 0 in the UV account to set system-wide default SETPTR parameters.

To specify systemwide defaults, first use a SETPTR 0 command to set the defaults you want to use, then use the SETPTR.DEFAULT command.

If you do not set parameters for logical print channel 0 before you use SETPTR.DEFAULT, SETPTR.DEFAULT sets systemwide defaults to the initial system defaults.

Default SETPTR parameters are always reset to the initial system defaults after you reboot your system.

Example

In the following example, SETPTR sets parameters for logical print channel 0. It sets line length to 80, specifies form LW (laser writer), and suppresses the banner page and the end-of-job blank page. The SETPTR.DEFAULT command makes these parameters the system-wide defaults.

```
>SETPTR 0,80,,,,,FORM LW,NOHEAD,NOEJECT,BRIEF
>SETPTR.DEFAULT
>
```

SETUP.DEMO.SCHEMA

Use SETUP.DEMO.SCHEMA to convert the account you are logged in to a UniVerse SQL schema called DEMO_ *username*. You must have DBA privilege to convert an account into a schema.

Syntax

SETUP.DEMO.SCHEMA *username*

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>username</i>	The user name whose account you want to convert into a schema. This user is the owner of the new schema.

SETUP.DEMO,SCHEMA Parameter

Description

This command lets the DBA set up an SQL schema for another user who wants to run [MAKE.DEMO.TABLES](#). SETUP.DEMO.SCHEMA registers *username* as an SQL user (if not already).



SH

Use SH to invoke a UNIX Bourne shell (*sh*) from within UniVerse.

Note: This command is not supported on Windows platforms.

Syntax

SH [`-c "command" | script`]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>command</i>	The command text. It must be preceded by the flag <code>-c</code> . To avoid confusion, enclose <i>command</i> in quotation marks.
<i>script</i>	The pathname of a UNIX shell script to execute. The shell script file does not need execute privileges.

SH Parameters

Description

Once you invoke a Bourne shell, you can execute any UNIX command. You can also give SH an argument to execute a single command or a shell script without exiting UniVerse. To return to the UniVerse prompt, enter **exit** or press Ctrl-D.

See your UNIX documentation for a discussion of Bourne shell features.

Use the CSH command to execute UNIX C shell (*csh*) commands.

Example

This example invokes a Bourne shell and runs the UNIX command `ls` to list the contents of the current directory:

```
>SH -c"ls -lt"
total 707
drwxrwxrwx  2 susan  acctg          512 Jul  6 11:43 I_CREDITORS
-rw-rw-r--  1 susan  acctg        1536 Jul  6 11:43 CREDITORS
-rw-rw-r--  1 susan  acctg        2048 Jul  6 11:42 D_CREDITORS
-rw-r--r--  1 susan  acctg       49152 Jul  6 11:41 VOC
drwxrwxrwx  2 susan  acctg          512 Jul  2 16:55 I_ORDERS
-rw-r--r--  1 susan  acctg        3072 Jul  2 16:55 D_ORDERS
drwxrwxr-x  2 susan  acctg          512 Jul  2 16:43
LONG.OVERDUE
-rw-rw-r--  1 susan  acctg        4096 Jul  2 16:41 D_LONG.OVERDUE
-rw-rw-r--  1 susan  acctg        2048 Jul  2 16:40 D_OVERDUE
-rw-rw-r--  1 susan  acctg        1536 Jul  2 16:40 OVERDUE
drwxrwxrwx  2 susan  acctg          512 Jul  2 16:07 CUSTOMERS
-rw-r--r--  1 susan  acctg        3072 Jul  2 16:06 D_CUSTOMERS
```

SHUTDOWN.RECOVERY

Use SHUTDOWN.RECOVERY to disable the transaction logging system if it is enabled or suspended. You must be a UniVerse Administrator logged in to the UV account to use SHUTDOWN.RECOVERY.

Syntax

SHUTDOWN.RECOVERY [INFORM]

Parameter

The following table describes the parameter of the syntax:

Parameter	Description
INFORM	Displays messages during the shutdown process.

SHUTDOWN.RECOVER Parameter

Description

Use SHUTDOWN.RECOVERY in extreme situations to disable the transaction logging system. For example, if your tape unit is unserviceable and all log files are full but processing must continue, SHUTDOWN.RECOVERY permits nontransactional updates to recoverable files without logging updates. You can reenableView transaction logging with [ENABLE.RECOVERY](#).



Warning: *Because updates to recoverable files are not logged after you execute SHUTDOWN.RECOVERY, the point of disablement is the latest point to which you can consistently recover your UniVerse files.*

If transaction logging is currently full or suspended, SHUTDOWN.RECOVERY starts the log daemon to disable logging properly. While transaction logging is in the disabled state, programs that request writes to the log file fail.

Example

This example disables transaction logging:

```
>SHUTDOWN.RECOVERY
```

```
Request to Shutdown Logging Subsystem made at 12:47:59 on 01 OCT  
1996. You can use the 'Display logging state' menu to verify the  
current state of the logging subsystem.
```

SLEEP

Use SLEEP to suspend a process for a specific length of time or until a specific time of day. SLEEP is useful in a paragraph that is executed as a phantom process to specify the time to begin the process.

Syntax

SLEEP [*seconds* | *hh:mm[:ss]*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>seconds</i>	Specifies the number of seconds to suspend processing. If you omit seconds, the default of 1 second is used.
<i>hh:mm</i>	Specifies the hour and minute when you want the process to restart. <i>hh</i> is in 24-hour format. You can also specify the seconds with <i>:ss</i> .

SLEEP Parameters

Description

To suspend a process for a specified amount of time, use the command in the following format:

SLEEP [*seconds*]

To suspend a process until a specific time of day, use SLEEP in the following format:

SLEEP *hh:mm[:ss]*

Remember to use the 24-hour format. If you want to specify 7:30 p.m., you must give the time as 19:30.

Example

This example suspends the current process until 3:30 p.m.:

```
>SLEEP 15:30
```

SORT

Use SORT to display selected data from a file in sorted order.

Syntax

SORT [DICT | USING [DICT] *dictname*] *filename* [*records* | FROM *n*]
[*selection*] [*output.limiter*] [*sort*] [*output*] [*report.qualifiers*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Lists records in the file dictionary of <i>filename</i> . If you do not specify DICT, records in the data file are listed.
USING [DICT] <i>dictname</i>	If DICT is not specified, uses <i>dictname</i> as the dictionary of <i>filename</i> . If DICT is specified, the dictionary of <i>dictname</i> is used as the dictionary of <i>filename</i> .
<i>filename</i>	The file whose records you want to list. You can specify <i>filename</i> anywhere in the sentence. SORT uses the first word in the sentence that has a file descriptor in the VOC file as the file name.
<i>records</i>	Specifies the records to include in the list. You can specify as many record IDs as you want, separated by spaces. Enclose record IDs in single quotation marks to ensure that Retrieve does not interpret them as field names or keywords.
FROM <i>n</i>	Specifies the records whose record IDs are stored in select list <i>n</i> .
<i>selection</i>	A selection expression specifying the conditions that data in a record must meet for the record to be selected. A selection expression begins with the keyword WITH. For syntax details, see the WITH keyword in Chapter 2, “ UniVerse Keywords .”
<i>output.limiter</i>	An expression specifying the conditions that values in multivalued fields must meet for those values to be output. An output-limiting expression begins with the keyword WHEN. For syntax details, see the WHEN keyword in Chapter 2, “ UniVerse Keywords .”

SORT Parameters

Parameter	Description																					
<i>sort</i>	A sort expression that specifies the type of sort and the field to sort on. The syntax is as follows: <table><tr><td>BY <i>field</i></td><td>Sort on <i>field</i> in ascending order.</td></tr><tr><td>BY.DSND <i>field</i></td><td>Sort on <i>field</i> in descending order.</td></tr><tr><td>BY.EXP <i>field</i></td><td>Explode multivalues in <i>field</i> and <i>sort</i> in ascending order.</td></tr><tr><td>BY.EXP.DSND <i>field</i></td><td>Explode multivalues in <i>field</i> and <i>sort</i> in descending order.</td></tr></table> For more information about sort expressions, see Chapter 2, “UniVerse Keywords.” When NLS locales are enabled, SORT uses the Collate convention of the current locale to determine the collating order for any existing sort expression. If an index has its own Collate convention defined, SORT uses it instead of the current locale definition. For more information, see the <i>UniVerse NLS Guide</i>.	BY <i>field</i>	Sort on <i>field</i> in ascending order.	BY.DSND <i>field</i>	Sort on <i>field</i> in descending order.	BY.EXP <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in ascending order.	BY.EXP.DSND <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in descending order.													
BY <i>field</i>	Sort on <i>field</i> in ascending order.																					
BY.DSND <i>field</i>	Sort on <i>field</i> in descending order.																					
BY.EXP <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in ascending order.																					
BY.EXP.DSND <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in descending order.																					
<i>output</i>	Specifies the fields whose data you want to list. The fields must be defined in the file dictionary. If you are listing dictionary records, the fields must be defined in the DICT.DICT file. You can precede a field name with one field modifier. Field modifiers are: <table><tr><td>AVG</td><td>ENUM</td><td>PCT</td></tr><tr><td>BREAK.ON</td><td>MAX</td><td>TOTAL</td></tr><tr><td>BREAK.SUP</td><td>MIN</td><td>TRANSPORT</td></tr><tr><td>CALC</td><td></td><td></td></tr></table> You can follow a field name with one or more field qualifiers. Field qualifiers are: <table><tr><td>AS</td><td>COL.HDG</td><td>FMT</td></tr><tr><td>ASSOC</td><td>CONV</td><td>MULTI.VALUE</td></tr><tr><td>ASSOC.WITH</td><td>DISPLAY.LIKE</td><td>SINGLE.VALUE</td></tr></table> For information about these keywords and their synonyms, see Chapter 2, “UniVerse Keywords.” If you do not specify <i>output</i>, fields specified in the @ phrase are listed. If there is no @ phrase in the file dictionary, only record IDs are listed.	AVG	ENUM	PCT	BREAK.ON	MAX	TOTAL	BREAK.SUP	MIN	TRANSPORT	CALC			AS	COL.HDG	FMT	ASSOC	CONV	MULTI.VALUE	ASSOC.WITH	DISPLAY.LIKE	SINGLE.VALUE
AVG	ENUM	PCT																				
BREAK.ON	MAX	TOTAL																				
BREAK.SUP	MIN	TRANSPORT																				
CALC																						
AS	COL.HDG	FMT																				
ASSOC	CONV	MULTI.VALUE																				
ASSOC.WITH	DISPLAY.LIKE	SINGLE.VALUE																				

SORT Parameters (Continued)

Parameter	Description			
<i>report.qualifiers</i>				
One or more of the following keywords:				
	COL.HDR.SUPP	FIRST	ID.ONLY	ONLY
	COL.SPCS	FOOTING	ID.SUP	SAMPLE
	COL.SUP	GRAND.TOTAL	LPTR	SAMPLED
	COUNT.SUP	HDR.SUP	MARGIN	SUPP
	DBL.SPC	HEADING	NO.SPLIT	VERT
	DET.SUP		NOPAGE	
These keywords modify the report format. For information about them and their synonyms, see Chapter 2, “ UniVerse Keywords .”				
If you do not specify any report qualifiers, SORT produces a report with:				
n	UniVerse default heading and footing			
n	Single spacing between records			
n	Column headings on every page			
n	No control breaks			
n	Paged output to the screen			

SORT Parameters (Continued)

Description

SORT lists the selected records in the order specified by the sort expression. If there is no sort expression, it lists them in record ID order.

Specifying Output

In an *output* specification you can specify as many field names as you want, separated by spaces. You can specify field names explicitly or as a phrase containing field names. You can use field names and phrases together in the same output specification. Data in the specified fields is listed in columns in the order in which field names appear in the sentence or @ phrase.

Using Field Expressions

You can use field expressions in selection expressions, sort expressions, output specifications, and report clauses. A field expression can be a field name with or without field qualifiers, or it can be an EVAL expression. An EVAL expression is an I-type expression specified on the command line, introduced by the keyword EVAL.

Suppressing Record IDs

By default, SORT displays record IDs in the first column of output. Use the ID.SUP keyword to suppress the display of record IDs.

SORT.ITEM

Use SORT.ITEM to display a complete listing of selected records.

Syntax

SORT.ITEM [DICT | USING [DICT] *dictname*] *filename* [*records* | FROM *n*]
[*selection*] [*sort*] [*report.qualifiers*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Lists records in the file dictionary of <i>filename</i> . If you do not specify DICT, records in the data file are listed.
USING [DICT] <i>dictname</i>	If DICT is not specified, uses <i>dictname</i> as the dictionary of <i>filename</i> . If DICT is specified, the dictionary of <i>dictname</i> is used as the dictionary of <i>filename</i> .
<i>filename</i>	The file whose records you want to list. You can specify <i>filename</i> anywhere in the sentence. SORT.ITEM uses the first word in the sentence that has a file descriptor in the VOC file as the file name.
<i>records</i>	Specifies the records to include in the list. You can specify as many record IDs as you want, separated by spaces. Enclose record IDs in single quotation marks to ensure that Retrieve does not interpret them as field names or keywords.
FROM <i>n</i>	Specifies the records whose record IDs are stored in select list <i>n</i> .
<i>selection</i>	A selection expression specifying the conditions that data in a record must meet for the record to be selected. A selection expression begins with the keyword WITH. For syntax details, see the WITH keyword in Chapter 2, “UniVerse Keywords.”
<i>sort</i>	A sort expression that specifies the type of sort and the field to sort on. The syntax is as follows:

SORT.ITEM Parameters

Parameter	Description																
	BY <i>field</i> Sort on <i>field</i> in ascending order.																
	BY.DSND <i>field</i> Sort on <i>field</i> in descending order.																
	BY.EXP <i>field</i> Explode multivalues in <i>field</i> and <i>sort</i> in ascending order.																
	BY.EXP.DSND <i>field</i> Explode multivalues in <i>field</i> and <i>sort</i> in descending order.																
	For more information about sort expressions, see Chapter 2, “UniVerse Keywords.” When NLS locales are enabled, SORT.ITEM uses the Collate convention of the current locale to determine the collating order for any existing sort expression. If an index has its own Collate convention defined, SORT.ITEM uses it instead of the current locale definition. For more information, see the <i>UniVerse NLS Guide</i>.																
<i>report.qualifiers</i>	One or more of the following keywords: <table><tr><td>COL.HDR.SUPP</td><td>HDR.SUP</td><td>LPTR</td><td>SAMPLE</td></tr><tr><td>DBL.SPC</td><td>HEADING</td><td>MARGIN</td><td>SAMPLED</td></tr><tr><td>FIRST</td><td>ID.SUP</td><td>NOPAGE</td><td>SUPP</td></tr><tr><td>FOOTING</td><td></td><td></td><td></td></tr></table> These keywords modify the report format. For information about them and their synonyms, see Chapter 2, “UniVerse Keywords.” If you do not specify any report qualifiers, SORT.ITEM produces a report with: n UniVerse default heading n Single spacing between records n Paged output to the screen	COL.HDR.SUPP	HDR.SUP	LPTR	SAMPLE	DBL.SPC	HEADING	MARGIN	SAMPLED	FIRST	ID.SUP	NOPAGE	SUPP	FOOTING			
COL.HDR.SUPP	HDR.SUP	LPTR	SAMPLE														
DBL.SPC	HEADING	MARGIN	SAMPLED														
FIRST	ID.SUP	NOPAGE	SUPP														
FOOTING																	

SORT.ITEM Parameters (Continued)

Description

SORT.ITEM ignores RetrieVe field output specifications.

`SORT.ITEM` is like the Pick version of the `COPY` command (using the P or T option), but it lets you specify selection criteria and headings and footings.

SORT.LABEL

Use SORT.LABEL to specify a format suitable for mailing labels and other specialized block listings. The report lists the fields for each record in a block, with headings in the leftmost column. The labels are in sorted order.

Syntax

```
SORT.LABEL [ DICT | USING [ DICT ] dictname ] filename  
[ records | FROM n ] [ selection ] [ output.limiter ] [ sort ] [ output ]  
[ report.qualifiers ]
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Lists records in the file dictionary of <i>filename</i> . If you do not specify DICT, records in the data file are listed.
USING [DICT] <i>dictname</i>	If DICT is not specified, uses <i>dictname</i> as the dictionary of <i>filename</i> . If DICT is specified, the dictionary of <i>dictname</i> is used as the dictionary of <i>filename</i> .
<i>filename</i>	The file whose records you want to list. You can specify <i>filename</i> anywhere in the sentence. SORT.LABEL uses the first word in the sentence that has a file descriptor in the VOC file as the filename.
<i>records</i>	Specifies the records to include in the list. You can specify as many record IDs as you want, separated by spaces. Enclose record IDs in single quotation marks to ensure that Retrieve does not interpret them as field names or keywords.
FROM <i>n</i>	Specifies the records whose record IDs are stored in select list <i>n</i> .
<i>selection</i>	A selection expression specifying the conditions that data in a record must meet for the record to be selected. A selection expression begins with the keyword WITH. For syntax details, see the WITH keyword in Chapter 2, “UniVerse Keywords.”

SORT.LABEL Parameters

Parameter	Description								
<i>output.limiter</i>	An expression specifying the conditions that values in multivalued fields must meet for those values to be output. An output-limiting expression begins with the keyword WHEN. For syntax details, see the WHEN keyword in Chapter 2, “ UniVerse Keywords. ”								
<i>sort</i>	A sort expression that specifies the type of sort and the field to sort on. The syntax is as follows: <table> <tr> <td>BY <i>field</i></td><td>Sort on <i>field</i> in ascending order.</td></tr> <tr> <td>BY.DSND <i>field</i></td><td>Sort on <i>field</i> in descending order.</td></tr> <tr> <td>BY.EXP <i>field</i></td><td>Explode multivalues in <i>field</i> and <i>sort</i> in ascending order.</td></tr> <tr> <td>BY.EXP.DSND <i>field</i></td><td>Explode multivalues in <i>field</i> and <i>sort</i> in descending order.</td></tr> </table> For more information about sort expressions, see Chapter 2, “UniVerse Keywords.” When NLS locales are enabled, SORT.LABEL uses the Collate convention of the current locale to determine the collating order for any existing sort expression. If an index has its own Collate convention defined, SORT.LABEL uses it instead of the current locale definition. For more information, see the <i>UniVerse NLS Guide</i>.	BY <i>field</i>	Sort on <i>field</i> in ascending order.	BY.DSND <i>field</i>	Sort on <i>field</i> in descending order.	BY.EXP <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in ascending order.	BY.EXP.DSND <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in descending order.
BY <i>field</i>	Sort on <i>field</i> in ascending order.								
BY.DSND <i>field</i>	Sort on <i>field</i> in descending order.								
BY.EXP <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in ascending order.								
BY.EXP.DSND <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in descending order.								
<i>output</i>	Specifies the fields whose data you want to list. The fields must be defined in the file dictionary. If you are listing dictionary records, the fields must be defined in the DICT.DICT file. If you do not specify <i>output</i>, fields specified in the @ phrase are listed. If there is no @ phrase in the file dictionary, only record IDs are listed.								

SORT.LABEL Parameters (Continued)

Parameter	Description
<i>report.qualifiers</i>	One or more of the following keywords: COL.HDR.SUPP FOOTING ID.SUP SAMPLE COUNT.SUP HDR.SUP LPTR SAMPLED DBL.SPC HEADING NOPAGE SUPP FIRST ID.ONLY ONLY These keywords modify the report format. For information about them and their synonyms, see Chapter 2, “UniVerse Keywords.” If you do not specify any report qualifiers, SORT.LABEL produces a report with: - n UniVerse default heading and footing - n Paged output to the screen

SORT.LABEL Parameters (Continued)

Description

After you enter a SORT.LABEL command, the following prompt appears:

COUNT, ROWS, SKIP, INDENT, SIZE, SPACE [,C] ?

Enter numeric values for each of these specifications (except C) to produce the formatted report. The following list describes what to enter:

Specification	Description
COUNT	The number of labels across the page or screen.
ROWS	The number of lines (fields) displayed or printed per label.
SKIP	The number of blank lines vertically between labels.
INDENT	The number of indented spaces from left margin to the first column of labels. Can be zero.
SIZE	The maximum number of characters in any display field.
SPACE	The number of blank spaces horizontally between labels.
C	Do not print empty fields. If you do not specify C, empty fields are printed as a series of blanks.

SORT.LABEL Specifications

After you enter these specifications, a series of prompts requests headers for each row in a label. You can enter a header or press Return to specify no header.

Row 1 header?**NAME**

In the report these headers appear in the left margin for each set of labels. If INDENT is zero, the command does not prompt for row headers. Be sure to specify an indent area wide enough for the headers.

The total width specifications cannot exceed the capability of the output device, whether it is the terminal screen or the printer. Use the following formula to calculate the total width:

$$\text{INDENT} + \text{COUNT}(\text{SIZE} + \text{SPACE})$$

To produce a continuous report without page breaks, use the [COL.HDR.SUPP](#) keyword. This keyword also suppresses the header at the top of the report.

Example

This example creates labels made up of the record ID, the first name, and the last name. The record IDs are listed by default. The second line is the format specification. COUNT specifies three labels across the page, with each label made up of three fields (ROWS). Two blank lines vertically separate each row of labels (SKIP), the first column of labels is indented 10 spaces from the left (INDENT), each display field is 15 characters wide (SIZE), and 2 blank spaces horizontally separate each column of labels (SPACE).

Because INDENT specifies 10 spaces, SORT.LABEL prompts the user to enter a row header for each row.

```
>SORT.LABEL SUN.MEMBER FNAME LNAME
COUNT, ROWS, SKIP, INDENT, SIZE, SPACE [ ,C ] ? 3,3,2,10,15,2
Row 1 header ID
Row 2 header NAME
Row 3 header <Return>
```

This is the output:

SORT.LABEL SUN.MEMBER FNAME LNAME 11:09:10am 20 Oct 1995 PAGE 1			
ID	2342	3452	4102
NAME	RALPH	JANE	LESLIE
	ADDAMS	SAMUEL	BROWN
ID	4108	4309	4439

NAME	HILLARY HENDERSON	EDGAR WILLIAMS	DON ALISON
ID	5205	5390	6100
NAME	ALICE CRATCHETT	ALICE MIX	BOB MASTERS
ID	6203	6783	7100
NAME	SAM YORK	DAVID HALE	ALICE WILLIAMS
ID	7505		
NAME	HARRY EDWARDS		

13 records listed.

SP.ASSIGN (UNIX)

Use SP.ASSIGN to set the line printer spooler options for each of the 256 logical print channels. These changes remain set until you use SP.ASSIGN again or use LOGOUT or QUIT. If you specify no options, SP.ASSIGN uses default settings.

Syntax

```
SP.ASSIGN [=formname] [copies] [?] [D] [{ F | Q } form] [Runit] [H] [S]
[O] [T]
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>formname</i>	Sets the form name of <i>unit</i> to <i>formname</i> . You can use alphabetic and numeric characters to specify <i>formname</i> .
<i>copies</i>	The number of copies to print of each job.
<i>?</i>	Lists the current status. If you specify only <i>?</i> , no unspecified options are changed to their defaults. As the last character of a UniVerse sentence, <i>?</i> puts the sentence on the sentence stack without executing it. When you specify <i>?</i> at the end of the command line, be sure to enter an extra space after the <i>?</i> to prevent the sentence from being put on the sentence stack unexecuted.
<i>D</i>	Resets unspecified options to their default values.
<i>Fform</i>	Sets the form number of <i>unit</i> to the number specified by <i>form</i> . Same as <i>Qform</i> .
<i>Qform</i>	Same as <i>Fform</i> .
<i>Runit</i>	Indicates the logical print channel number.
<i>H</i>	Retains jobs in the spool queue after they are printed the first time.

SP.ASSIGN Parameters

Parameter	Description
	You can reprint held jobs using <i>usm</i> with the <i>-r</i> option or using the SP.EDIT command. Reprinting the job does not remove it from the spool queue. You can remove held jobs from the spool queue using the <i>-k</i> option of <i>usm</i> or using the SP.EDIT command.
S	Suppresses all output. If you specify S with the H option, jobs are sent to the spool queue in the hold state, not printed.
O	Causes the spool file to remain open after the report. Subsequent reports are appended to the same spool file.
T	Sets the form name to TAPE.
SP.ASSIGN Parameters (Continued)	

SP.ASSIGN (Windows Platforms)

Use SP.ASSIGN to set the line printer spooler options for Windows platforms. These changes remain set until you use SP.ASSIGN again or use LOGOUT or QUIT. If you specify no options, SP.ASSIGN uses default settings.

Syntax

SP.ASSIGN [*copies*] [{ F | Q } *form*] [S] [O] [T]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>copies</i>	The number of copies to print of each job. In GDI mode, the number of copies is passed to the printer driver. In raw mode, this number is passed to Windows as a parameter for the print processor.
<i>Fform</i>	Sets the form number of <i>unit</i> to the number specified by <i>form</i> . Same as <i>Qform</i> .
<i>Qform</i>	Same as <i>Fform</i> .
S	Suppresses all output.
O	Causes the spool file to remain open after the report. Subsequent reports are appended to the same spool file.
T	Sets the form name to TAPE.

SP.ASSIGN Parameters

SP.EDIT (UNIX)

Use SP.EDIT to select held spool files from the spool queue and send them to either the terminal or the printer.

Syntax

SP.EDIT [*entry*[-*entry*]] [*Fform*[-*form*]] [*L*] [*R*] [*U*] [*MD*] [*MS*] [*O*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>entry</i>	The number of the entry to manipulate.
<i>Fform</i>	Specifies only entries from the specified form numbers.
<i>L</i>	Lists the first 512 bytes of each spool file before the showing the <code>Display</code> prompt. This option overrides the action of the <code>MD</code> and <code>MS</code> options.
<i>R</i>	Causes spooled files in the hold state to use the current SETPTR (UNIX) characteristics. Without this option, spooled files use the options they were initially spooled with. This changes the default options for the job.
<i>U</i>	Looks at files belonging to other users. You must be a UniVerse Administrator to use this option.
<i>MD</i>	Deletes all selected files.
<i>MS</i>	Spools all selected files.
<i>O</i>	Looks at active and waiting files and files in a held (or paused) state.

SP.EDIT Parameters

Description

After you enter SP.EDIT, the `Display` prompt appears. SP.EDIT prints spooler statistics for the current file on the terminal before showing the `Display` prompt. The format of the spooler statistics follow:

Entry	#	Status	:	Size	:
Name	:	Banner	:	Message	:
Priority	:	Copies	:	Delay	:
Pages	:	Eject	:	Delete	:
Form	:	Printer	:	Options	:

These fields display the following statistics:

Column Heading	Description
Entry	The job number in the spool queue.
Name	The login ID of the user who sent the job.
Priority	The priority of the job. A lower number indicates higher priority.
Pages	The number of pages to print. This field can contain a single page number, a range of page numbers separated by a dash or a comma, or the word <code>All</code> .
Form	The form on which to print the job.
Status	The state of the job. This can be <code>Active</code> , <code>Wait</code> , or <code>Hold</code> . An asterisk (<code>*</code>) in this field indicates that the file requeues after printing.
Banner	The main text printed on the header page.
Copies	The number of copies to print.
Eject	Eject extra page after the job (<code>Yes</code> or <code>No</code>).
Printer	The name of the printer. If no printer is specified for the job, this field contains <code>[Any]</code> .
Size	The size of the file in bytes.

SP.EDIT Display

Column Heading	Description
Message	The secondary text printed under the banner on the header page.
Delay	The number of seconds to delay before printing.
Delete	Delete the temporary printing source file after printing (Yes or No). This is normally Yes.
Options	The printing options set for this file. None indicates no options. LNUM indicates that line numbering is set. FTN indicates that the job is passed through a FORTRAN filter before printing.

SP.EDIT Display (Continued)

The Display Prompt

SP.EDIT prompts you to enter display specifications:

Display (Y/N/S/D/X/<CR>) ?

The responses to the Display prompt are as follows:

Y Displays first 512 bytes of the file.

- For files in the active state, SP.EDIT displays the following prompt:

Kill (Y/N=CR) ?

The responses to the Kill prompt are as follows:

Y Displays the Hold in Queue prompt:

Hold in Queue (Y=CR/N) ?

The responses to the Hold in Queue prompt are as follows:

Y (or ENTER) Leaves the job in the spool queue in the hold state and activate requeuing for the file.

N Removes the job from the spool queue.

N Displays the Display prompt for the next file.

- For files in the wait state, SP.EDIT displays the following prompt:

Hold in Queue (Y/N=CR) ?

The responses to the Hold in Queue prompt are as follows:

Y Leaves the job in the spool queue in the hold state and activates requeuing for the file.

N (or Return) Displays the Delete prompt.

- For files in all other states, SP.EDIT displays the Spool prompt.

N Does not display the first 512 bytes of the file, but proceeds as with Y.

S Displays the Spool prompt.

D Displays the Delete prompt.

X Exits SP.EDIT and returns to the command line.

If you press ENTER at the Display prompt, SP.EDIT displays the Display prompt for the next file.

The Spool Prompt

If you enter **S** at the Display prompt, SP.EDIT displays the following prompt:

Spool (Y/N=CR) ?

The responses to the Spool prompt are as follows:

Y SP.EDIT displays either a Hold in Queue prompt or a Pages to Output prompt.

- For files that are not set to requeue after printing, SP.EDIT displays the following prompt:

Hold in Queue (Y/N=CR) ?

The responses to the Hold in Queue prompt are as follows:

Y Sets the file to requeue after printing and displays the Pages to Output prompt.

N (or ENTER) Displays the Pages to Output prompt.

- For files that are set to requeue after printing and files that have been processed by `Hold in Queue`, SP.EDIT displays the following prompt:

Pages to Output :

Enter a page number, a starting and ending page number separated by a dash (–) or a comma (,), or **a11** to indicate which pages to print. A Return does not modify the current status.

N (or ENTER) displays the Delete prompt.

The Delete Prompt

If you enter **D** at the `Display` prompt or **N** at the `Spool` prompt, SP.EDIT displays the following prompt:

Delete (Y/N=CR) ?

The responses to the Delete prompt are as follows:

Y Removes the job from the spool queue, then displays the `Display` prompt for the next file.

N (or Return) Displays the `Display` prompt for the next file.

SP.EDIT (Windows Platforms)

Use SP.EDIT to administer print jobs in the Windows printer queues.

Syntax

SP.EDIT [*entry* [*-entry*]] [*Fform* [*-form*]] [*L*] [*R*] [*U*] [*MD*] [*MS*] [*O*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>entry</i>	The number of the entry to manipulate.
<i>Fform</i>	Specifies only entries from the specified form numbers.
<i>L</i>	Displays the first 512 bytes of a print job, instead of prompting for actions. This qualifier only works for print jobs created in Windows raw mode. This option overrides the action of the MD and MS options.
<i>R</i>	Causes spooled files in the hold state to use the current SETPTR (Windows Platforms) characteristics. Without this option, spooled files use the options they were initially spooled with. This changes the default options for the job.
<i>U</i>	Looks at files belonging to other users. You must be a UniVerse Administrator to use this option.
<i>MD</i>	Deletes all selected files.
<i>MS</i>	Spools all selected files.
<i>O</i>	Looks at active and waiting files as well as files in a held (or paused) state.

SP.EDIT Parameters

Description

After you enter SP.EDIT, the `Display` prompt appears. SP.EDIT prints spooler statistics for the current file on the terminal before displaying the `Display` prompt. The format of the spooler statistics follow:

Entry	#	Status	:	Size	:
Jobname	:	Username	:	Printer	:
Priority	:	Copies	:	Delay	:
Form	:	Created	:		

These fields display the following statistics:

Column Heading	Description
Entry	The job number in the spool queue.
Jobname	The name of the print job, taken from the BANNER attribute when the job was created. If a banner has not been set, the job name defaults to UniVerse.
Priority	The priority of the job, in UniVerse format. A lower number indicates higher priority.
Form	The form name specified for the job.
Status	The current state of the job. If the job is simply waiting to be printed, no status is shown; otherwise, it can be <code>Paused</code> or <code>Printing</code> .
Username	The user name of the user who sent the print job.
Copies	The number of copies to print.
Created	The time and date the spooled file was created.
Size	The size of the file in bytes.
Printer	The name of the printer.
Delay	The start time specified for printing.

SP.EDIT Display

The Display Prompt

SP.EDIT prompts you to enter display specifications:

Display (Y/N/S/D/X/<CR>) ?

The responses to the Display prompt are as follows:

Y Displays first 512 bytes of the file, if the print job was created in RAW mode.

- For files already printing, SP.EDIT displays the following prompt:

Kill (Y/N=CR) ?

The responses to the Kill prompt are as follows:

Y Displays the Are you sure prompt:

Are you sure (Y=CR/N) ?

The responses to the Are you sure prompt are as follows:

Y Cancels printing of this job and removes it from the queue.

N (or ENTER) Goes to the Display prompt for the next file.

N Displays the Display prompt for the next file.

- For files waiting to be printed, SP.EDIT displays the following prompt:

Hold in Queue (Y/N=CR) ?

The responses to the Hold in Queue prompt are as follows:

Y Pauses the print job.

N (or ENTER) Displays the Delete prompt.

- For files that have been paused, or that are in any other state, SP.EDIT displays the Spool prompt.

N Does not display the first 512 bytes of the file, but proceeds as with Y.

S Does not display the first 512 bytes of the file, but proceeds as with Y.

D Displays the Delete prompt.

X Exits SP.EDIT and returns to the command line.

If you press ENTER at the Display prompt, SP.EDIT displays the Display prompt for the next file.

The Spool Prompt

If you enter **Y** at the `Display` prompt, and the file is currently paused or in another state (other than printing), SP.EDIT displays the following prompt:

Spool (Y/N=CR) ?

The responses to the `Spool` prompt are as follows:

Y Resumes the print job.

N (or ENTER) Displays the `Delete` prompt.

The Delete Prompt

If you enter **D** at the `Display` prompt or **N** at the `Spool` prompt, SP.EDIT displays the following prompt:

Delete (Y/N=CR) ?

The responses to the `Delete` prompt are as follows:

Y Cancels the print job, removing it from the printer queue and moves on to the next print job.

N (or ENTER) Displays the `Display` prompt for the next file.

SP.TAPE

Use SP.TAPE to print a report that has been stored on a spooled tape.

Syntax

SP.TAPE [*printer*] [*alignment*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>printer</i>	The name of a printer defined in the &DEVICE& file.
<i>alignment</i>	A number specifying how many lines to print at the beginning of the report so you can align your form in the printer. The number must be greater than or equal to 0. When you use <i>alignment</i>, SP.TAPE prints the number of lines from the beginning of the tape and asks if the form is aligned in the printer. It repeats this process until you indicate that the alignment is correct. Specify N if you do not want to align the printer. Do not use this parameter if you are printing to the system printer.

SP.TAPE Parameters

Description

You must be sure that the tape unit is assigned to you (see [ASSIGN](#)).

If you issue the SP.TAPE command with no qualifiers, SP.TAPE prompts you for the information.

You can create a report on tape using the ASSIGN command to assign a tape device as LPTR 0. Subsequent output to the printer will be spooled to the tape.

Examples

This example sends a print job stored on tape to the printer LP via the spooler. SP.TAPE prints the first 10 lines of the print job and asks if the alignment is correct. When the user enters **Y**, SP.TAPE spools the print job to the printer.

```
>SP.TAPE
Printer location = LP
Form alignment? (Number of lines or N) = 10
Is the form properly aligned? (Y/N) = Y

Spooling to printer "LP".
>
```

The next example specifies printer LP and no alignment (N) on the command line:

```
>SP.TAPE LP N

Spooling to printer "LP".
>
```

SPOOL

Use SPOOL to send records to the UniVerse spooler for printing, to examine the UniVerse spool queue, or to cancel print jobs in the spool queue. Users can list and cancel only their own print jobs. Printer group administrators can list and cancel all print jobs on printers in their printer group. A UniVerse Administrator can list and cancel all print jobs on all printers.

Syntax

```
SPOOL filename record [-AS alias] [ AT printer] [-COPIES n]  
[ FORM form] [-NOHEAD | -HEAD] [ PRINTER printer]  
  
SPOOL -LIST [[ AT printer] [-FORM form] [-PORTNO port#]  
[ PRINTER printer] [ USERNAME user]  
  
SPOOL -CANCEL job# [job#] ... [ AT printer] [-FORM form]  
[-PORTNO port#] [ PRINTER printer] [-RANGE start TO end]  
[-USERNAME user]
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>filename record</i>	The name of the file and the record ID of the record you want to print.
-LIST	Displays a list of jobs waiting to be printed. <i>printer</i> is case-sensitive. If you specify <i>printer</i> , only the jobs for that printer are listed; otherwise all jobs for all printers are listed.
-CANCEL	Deletes print jobs from the queue. <i>job#</i> is the number of the print job you want to cancel. You can specify multiple print job numbers, separated by spaces.
-AS	Prints <i>alias</i> instead of the filename on the flag page.
-AT	Specifies the name of the printer. <i>printer</i> is case-sensitive.

SPOOL Parameters

Parameter	Description
–COPIES	Specifies the number of copies to print.
–FORM	Specifies a special form. Before the file prints, the system operator must indicate the form is in the printer. <i>form</i> can be up to thirty-two characters long.
–HEAD	Specifies to print the flag page.
–NOHEAD	Suppresses the flag page.
–PORTNO	Specifies the port number of the user who submitted the print job.
–PRINTER	Specifies the name of the printer. <i>printer</i> is case-sensitive.
–RANGE	Specifies a range of print jobs to cancel. <i>start</i> and <i>end</i> are print job numbers. Any gaps in the range are ignored.
–USERNAME	Specifies the name of a user whose print jobs you want to list or cancel. <i>user</i> is case-sensitive.

SPOOL Parameters (Continued)

Description

The first available printer in the queue prints the records.

SPOOL can use an active select list.

SPOOL ignores all options if the records are spooled to the [&HOLD&](#) file.

To spool records to magnetic tape, assign a tape drive to a logical print channel and set up a form called TAPE. Use the following syntax to spool records to tape:

SPOOL *filename record* –FORM TAPE

Examples

The following examples show different ways you can cancel print jobs:

```
SPOOL -CANCEL -USERNAME julia
```

```
SPOOL -CANCEL -PRINTER LP0
```

```
SPOOL -CANCEL -PORTNO 45
```

```
SPOOL -CANCEL -FORM LANDSCAPE
```

```
SPOOL -CANCEL -RANGE 148 TO 156
```

The following examples show different ways you can list selected contents of the spooler queue:

```
SPOOL -LIST -USERNAME julia
```

```
SPOOL -LIST -PRINTER LP0
```

```
SPOOL -LIST -PORTNO 45
```

```
SPOOL -LIST -FORM LANDSCAPE
```

SPOOL (Windows Platforms)

Use SPOOL to send records to the system spooler for printing, to examine the spool queue, or to modify or cancel a print job that is in the spool queue. Users can list, modify, and cancel only their own print jobs. Printer group administrators can list, modify, and cancel all print jobs on printers in their printer group. A UniVerse Administrator can list, modify, and cancel all print jobs on all printers.

Syntax

SPOOL *filename record* [**-AS** *alias*] [**AT** *printer*] [**-COPIES** *n*]
[**FORM** *form*] [**-NOHEAD**]

SPOOL -LIST [*printer*]

SPOOL -CANCEL *job#* [*job#*] ...

SPOOL -MODIFY *job#* [{ **HOLD** | **NOHOLD** }] [**PRIORITY** *nnn*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>filename record</i>	The name of the file and the record ID of the record you want to print.
-LIST	Displays a list of jobs waiting to be printed. <i>printer</i> is case-sensitive. If you specify <i>printer</i> , only the jobs for that printer are listed; otherwise all jobs for all printers are listed.
-CANCEL	Deletes print jobs from the printer queue. <i>job#</i> is the number of the print job you want to cancel. You can specify multiple print job numbers separated by spaces.
-MODIFY	Changes the attributes of a print job already queued for printing. <i>job#</i> is the number of the print job you want to modify.
-AS	Prints <i>alias</i> instead of the filename on the flag page.

SPOOL Parameters

Parameter	Description
-AT	Specifies the name of the printer. <i>printer</i> is case-sensitive.
-COPIES	Specifies the number of copies to print.
-FORM	Specifies a special form. Before the file prints, the system operator must indicate the form is in the printer. <i>form</i> can be up to thirty-two characters long.
-NOHEAD	Suppresses the flag page.
HOLD	Suspends a print job. The suspended print job can be resumed by using the NOHOLD option, or by resuming the job from the Windows Print Manager.
NOHOLD	Turns off the HOLD option.
PRIORITY	Sets the print job's priority. The highest priority is 1 and the lowest is 255. The priority you enter is converted to a number within the range for Windows, which goes from 99 through 1. For more information, see the SETPTR (Windows Platforms) command.

SPOOL Parameters (Continued)

Description

The first available printer in the queue prints the records.

SPOOL can use an active select list.

SPOOL ignores all options if the records are spooled to the [&HOLD&](#) file.

To spool records to magnetic tape, assign a tape drive to a logical print channel and set up a form called TAPE. Use the following syntax to spool records to tape:

SPOOL *filename record* -FORM TAPE

SREFORMAT

Use SREFORMAT to create an inverted file. You can direct the output of the SREFORMAT command to another file or to magnetic tape.

Syntax

```
SREFORMAT [ DICT | USING [ DICT ] dictname ] filename
[ records | FROM n ] [ selection ] [ output.limiter ] [ sort ] output [ modifiers ] [ MTU
mtu ] [ BLK size ] [ labelopt ]
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Reformats records in the file dictionary of <i>filename</i> . If you do not specify DICT, records in the data file are reformatted.
USING [DICT] <i>dictname</i>	If DICT is not specified, uses <i>dictname</i> as the dictionary of <i>filename</i> . If DICT is specified, the dictionary of <i>dictname</i> is used as the dictionary of <i>filename</i> .
<i>filename</i>	The file whose records you want to reformat. You can specify <i>filename</i> anywhere in the sentence. SREFORMAT uses the first word in the sentence that has a file descriptor in the VOC file as the file name.
<i>records</i>	Specifies the records to reformat. You can specify as many record IDs as you want, separated by spaces. Enclose record IDs in single quotation marks to ensure that Retrieve does not interpret them as field names or keywords.
FROM <i>n</i>	Specifies the records whose record IDs are stored in select list <i>n</i> .
<i>selection</i>	A selection expression specifying the conditions that data in a record must meet for the record to be selected. A selection expression begins with the keyword WITH. For syntax details, see the WITH keyword in Chapter 2, “UniVerse Keywords.”

SREFORMAT Parameters

Parameter	Description
<i>output.limiter</i>	An expression specifying the conditions that values in multivalued fields must meet for those values to be output. An output-limiting expression begins with the keyword WHEN. For syntax details, see the WHEN keyword in Chapter 2, “ UniVerse Keywords. ”
SREFORMAT Parameters (Continued)	

Parameter	Description								
<i>sort</i>	A sort expression that specifies the type of sort and the field to sort on. The syntax is as follows: <table><tr><td>BY <i>field</i></td><td>Sort on <i>field</i> in ascending order.</td></tr><tr><td>BY.DSND <i>field</i></td><td>Sort on <i>field</i> in descending order.</td></tr><tr><td>BY.EXP <i>field</i></td><td>Explode multivalues in <i>field</i> and <i>sort</i> in ascending order.</td></tr><tr><td>BY.EXP.DSND <i>field</i></td><td>Explode multivalues in <i>field</i> and <i>sort</i> in descending order.</td></tr></table> For more information about sort expressions, see Chapter 2, “UniVerse Keywords.” When NLS locales are enabled, SREFORMAT uses the Collate convention of the current locale to determine the collating order for any existing sort expression. If an index has its own Collate convention defined, SREFORMAT uses it instead of the current locale definition. For more information, see the <i>UniVerse NLS Guide</i>.	BY <i>field</i>	Sort on <i>field</i> in ascending order.	BY.DSND <i>field</i>	Sort on <i>field</i> in descending order.	BY.EXP <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in ascending order.	BY.EXP.DSND <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in descending order.
BY <i>field</i>	Sort on <i>field</i> in ascending order.								
BY.DSND <i>field</i>	Sort on <i>field</i> in descending order.								
BY.EXP <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in ascending order.								
BY.EXP.DSND <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in descending order.								
<i>output</i>	Specifies the fields whose data you want to include. You must specify at least one field. The fields must be defined in the file dictionary. If you are listing dictionary records, the fields must be defined in the DICT.DICT file. You can precede a field name with the CALC field modifier. You cannot follow a field name with field qualifiers.								
<i>modifiers</i>	One or more of the following keywords: <table><tr><td>FIRST</td><td>ID.SUP</td><td>SAMPLE</td></tr><tr><td>ID.ONLY</td><td>ONLY</td><td>SAMPLED</td></tr></table> These keywords modify the report format. For information about them and their synonyms, see Chapter 2, “UniVerse Keywords.”	FIRST	ID.SUP	SAMPLE	ID.ONLY	ONLY	SAMPLED		
FIRST	ID.SUP	SAMPLE							
ID.ONLY	ONLY	SAMPLED							
MTU <i>mtu</i>	Specifies a drive other than 0. <i>mtu</i> indicates the mode (<i>m</i>), number of tracks (<i>t</i>), and tape unit number (<i>u</i>) of a tape drive. You can use the default for these values, 000, which indicates ASCII mode, 9-track tape, and unit number 0, otherwise SREFORMAT ignores the track specification. See the MTU keyword in Chapter 2, “UniVerse Keywords.”								

SREFORMAT Parameters (Continued)

Parameter	Description
<i>BLK size</i>	Specifies the block size if it is different from the default of 8192 bytes.
<i>labelopt</i>	One of the following: PICK.FORMAT (for Pick tape format) INFORMATION.FORMAT (for Prime INFORMATION tape format) REALITY.FORMAT (for REALITY tape format)
SREFORMAT Parameters (Continued)	

Description

After you enter an SREFORMAT command, the following prompt appears:

File Name =

Enter a valid UniVerse file name as a destination file, or enter the keyword TAPE (or BDE in IN2 flavor accounts). SREFORMAT directs output of the command either to the file or to magnetic tape, respectively. If you direct the output to another file, you must specify a file name other than the source file name. If you press ENTER at the File Name = prompt, no processing occurs.

When you send output to another file, SREFORMAT uses the first field defined by the output specification as the record ID in the new file. The other output fields become fields in the new file. The order of fields in the output specification of the SREFORMAT command determines the order of fields in the new file. The new file must already exist, and you must create new dictionary fields manually if you need them.

When you send output to magnetic tape, SREFORMAT writes one tape record for each reformatted record. Use the [T.ATT](#) command or the [BLK](#) keyword to specify tape record size. The default tape record size is 8192 bytes. The new record ID and the fields for the data record are concatenated, ending with a segment mark. They are either padded or truncated, depending on the size of the tape record. You can use [MTU](#), [BLK](#), and *labelopt* when directing output to tape.

SREFORMAT ignores control breaks and totals.

SSELECT

Use SSELECT to create a list of records that meet specified criteria, sorted by record ID. You can then use this list with other commands in the UniVerse system. Because the list contains the data or record IDs from selected records, it is called a select list.

Syntax

```
SSELECT [ DICT | USING [ DICT ] dictname ] filename [ records | FROM n ]  
[ selection ] [ output.limiter ] [ sort ] [ SAVING [ UNIQUE ] field ]  
[ NO.NULLS ] ] [ TO n ] [ report.qualifiers ]
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Selects records in the file dictionary of <i>filename</i> . If you do not specify DICT, records in the data file are selected.
USING [DICT] <i>dictname</i>	If DICT is not specified, uses <i>dictname</i> as the dictionary of <i>filename</i> . If DICT is specified, the dictionary of <i>dictname</i> is used as the dictionary of <i>filename</i> .
<i>filename</i>	The file whose records you want to select. You can specify <i>filename</i> anywhere in the sentence. SSELECT uses the first word in the sentence that has a file descriptor in the VOC file as the file name.
<i>records</i>	Specifies the records to include in the list. You can specify as many record IDs as you want, separated by spaces. Enclose record IDs in single quotation marks to ensure that Retrieve does not interpret them as field names or keywords.
FROM <i>n</i>	Specifies the records whose record IDs are stored in select list <i>n</i> .

SSELECT Parameters

Parameter	Description								
<i>selection</i>	A selection expression specifying the conditions that data in a record must meet for the record to be selected. A selection expression begins with the keyword WITH. For syntax details, see the WITH keyword in Chapter 2, “ UniVerse Keywords .”								
<i>output.limiter</i>	An expression specifying the conditions that values in multivalued fields must meet for those values to be output. An output-limiting expression begins with the keyword WHEN. For syntax details, see the WHEN keyword in Chapter 2, “ UniVerse Keywords .”								
<i>sort</i>	A sort expression that specifies the type of sort and the field to sort on. The syntax is as follows: <table> <tr> <td>BY <i>field</i></td><td>Sort on <i>field</i> in ascending order.</td></tr> <tr> <td>BY.DSND <i>field</i></td><td>Sort on <i>field</i> in descending order.</td></tr> <tr> <td>BY.EXP <i>field</i></td><td>Explode multivalues in <i>field</i> and <i>sort</i> in ascending order.</td></tr> <tr> <td>BY.EXP.DSND <i>field</i></td><td>Explode multivalues in <i>field</i> and <i>sort</i> in descending order.</td></tr> </table> For more information about sort expressions, see Chapter 2, “UniVerse Keywords.” When NLS locales are enabled, SSELECT uses the Collate convention of the current locale to determine the collating order for any existing sort expression. If an index has its own Collate convention defined, SSELECT uses it instead of the current locale definition. For more information, see the <i>UniVerse NLS Guide</i>.	BY <i>field</i>	Sort on <i>field</i> in ascending order.	BY.DSND <i>field</i>	Sort on <i>field</i> in descending order.	BY.EXP <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in ascending order.	BY.EXP.DSND <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in descending order.
BY <i>field</i>	Sort on <i>field</i> in ascending order.								
BY.DSND <i>field</i>	Sort on <i>field</i> in descending order.								
BY.EXP <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in ascending order.								
BY.EXP.DSND <i>field</i>	Explode multivalues in <i>field</i> and <i>sort</i> in descending order.								
SAVING	Specifies that the list be made up of field values instead of record IDs. Do not use a SAVING clause on multivalued fields.								
UNIQUE	Omits duplicates of the saved field from the select list.								
<i>field</i>	The name of a field whose values you want to constitute the select list.								
NO.NULLS	Omits empty string values in the saved field from the select list.								
TO <i>n</i>	The number to assign to the list, 0 through 10. If you do not specify a number, SSELECT creates select list 0.								
SSELECT Parameters (Continued)									

Parameter	Description
<i>report.qualifiers</i>	One or more of the following keywords: FIRST LPTR SAMPLE SAMPLED These keywords modify the report format. For information about them and their synonyms, see Chapter 2, “ UniVerse Keywords .”
SSELECT Parameters (Continued)	

STAT

Use STAT to total the numeric values in specified fields in a file. STAT also counts the selected records, and averages and totals the data contained in the fields.

Syntax

STAT [DICT | USING [DICT] *dictname*] *filename* [*records* | FROM *n*]
[*selection*] [*output.limiter*] [*fields*] [*report.qualifiers*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Processes records in the file dictionary of <i>filename</i> . If you do not specify DICT, records in the data file are processed.
USING [DICT] <i>dictname</i>	If DICT is not specified, uses <i>dictname</i> as the dictionary of <i>filename</i> . If DICT is specified, the dictionary of <i>dictname</i> is used as the dictionary of <i>filename</i> .
<i>filename</i>	The file whose records you want to process. You can specify <i>filename</i> anywhere in the sentence. STAT uses the first word in the sentence that has a file descriptor in the VOC file as the file name.
<i>records</i>	Specifies the records for which you want statistics. You can specify as many record IDs as you want, separated by spaces. Enclose record IDs in single quotation marks to ensure that Retrieve does not interpret them as field names or keywords.
FROM <i>n</i>	Specifies the records whose record IDs are stored in select list <i>n</i> .
<i>selection</i>	A selection expression specifying the conditions that data in a record must meet for the record to be selected. A selection expression begins with the keyword WITH. For syntax details, see the WITH keyword in Chapter 2, “UniVerse Keywords.”

STAT Parameters

Parameter	Description																				
<i>output.limiter</i>	An expression specifying the conditions that values in multi-valued fields must meet for those values to be output. An output-limiting expression begins with the keyword WHEN. For syntax details, see the WHEN keyword in Chapter 2, “ UniVerse Keywords .”																				
<i>fields</i>	The names of fields containing numeric data for which you want statistics. If a field contains nonnumeric data, STAT counts it but does not average or total the data.																				
<i>report.qualifiers</i>	One or more of the following keywords: <table><tr><td>COL.HDR.SUPP</td><td>FIRST</td><td>ID.SUP</td><td>SAMPLE</td></tr><tr><td>COL.SPCS</td><td>FOOTING</td><td>LPTR</td><td>SAMPLED</td></tr><tr><td>COL.SUP</td><td>HDR.SUP</td><td>MARGIN</td><td>SUPP</td></tr><tr><td>COUNT.SUP</td><td>HEADING</td><td>NOPAGE</td><td>VERT</td></tr><tr><td>DET.SUP</td><td>ID.ONLY</td><td>ONLY</td><td></td></tr></table> These keywords modify the report format. For information about them and their synonyms, see Chapter 2, “UniVerse Keywords.” If you do not specify any report qualifiers, STAT produces a report with: - n UniVerse default heading and footing - n Column headings on every page - n Paged output to the screen	COL.HDR.SUPP	FIRST	ID.SUP	SAMPLE	COL.SPCS	FOOTING	LPTR	SAMPLED	COL.SUP	HDR.SUP	MARGIN	SUPP	COUNT.SUP	HEADING	NOPAGE	VERT	DET.SUP	ID.ONLY	ONLY	
COL.HDR.SUPP	FIRST	ID.SUP	SAMPLE																		
COL.SPCS	FOOTING	LPTR	SAMPLED																		
COL.SUP	HDR.SUP	MARGIN	SUPP																		
COUNT.SUP	HEADING	NOPAGE	VERT																		
DET.SUP	ID.ONLY	ONLY																			

STAT Parameters (Continued)

STATUS

Use STATUS to display information about users and resources.

Syntax

STATUS [*option*] [NO.PAGE]

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
NO.PAGE	Suppresses automatic paging.

STATUS Parameter

option can be one of the following:

Option	Description
DISKS	Lists usage of all disks on the system.
ME	Lists all tasks related to your account. ME displays the same report as the LISTME command.
NETWORK	Lists the system name, node name, release number, version number, and machine name of your system.
USERS	Windows Platforms. USERS displays the same report as the LISTU command. UNIX. USERS lists all tasks for all users.
ALL	Lists all the status information.

STATUS Options

If you do not specify any option, STATUS produces a full report (same as ALL).

SUM

Use SUM to add the numeric values in specified fields of a file and list the totals at the end of each column of the report. You must include the name of at least one numeric field or a phrase name that includes a numeric field in the SUM sentence.

Syntax

```
SUM [ DICT | USING [ DICT ] dictname ] filename [ records | FROM n ]  
[ selection ] [ output.limiter ] [ fields ] [ report.qualifiers ]
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Processes records in the file dictionary of <i>filename</i> . If you do not specify DICT, records in the data file are processed.
USING [DICT] <i>dictname</i>	If DICT is not specified, uses <i>dictname</i> as the dictionary of <i>filename</i> . If DICT is specified, the dictionary of <i>dictname</i> is used as the dictionary of <i>filename</i> .
<i>filename</i>	The file whose records you want to total. You can specify <i>filename</i> anywhere in the sentence. SUM uses the first word in the sentence that has a file descriptor in the VOC file as the filename.
<i>records</i>	Specifies the records you want to total. You can specify as many record IDs as you want, separated by spaces. Enclose record IDs in single quotation marks to ensure that RetrieveVe does not interpret them as field names or keywords.
FROM <i>n</i>	Specifies the records whose record IDs are stored in select list <i>n</i> .
<i>selection</i>	A selection expression specifying the conditions that data in a record must meet for the record to be selected. A selection expression begins with the keyword WITH. For syntax details, see the WITH keyword in Chapter 2, “UniVerse Keywords.”

SUM Parameters

Parameter	Description																				
<i>output.limiter</i>	An expression specifying the conditions that values in multivalued fields must meet for those values to be output. An output-limiting expression begins with the keyword WHEN. For syntax details, see the WHEN keyword in Chapter 2, “UniVerse Keywords.”																				
<i>fields</i>	The names of fields containing numeric data to total. If a field contains nonnumeric data, SUM counts it but does not average or total the data.																				
<i>report.qualifiers</i>	One or more of the following keywords: <table><tr><td>COL.HDR.SUPP</td><td>FIRST</td><td>ID.SUP</td><td>SAMPLE</td></tr><tr><td>COL.SPCS</td><td>FOOTING</td><td>LPTR</td><td>SAMPLED</td></tr><tr><td>COL.SUP</td><td>HDR.SUP</td><td>MARGIN</td><td>SUPP</td></tr><tr><td>COUNT.SUP</td><td>HEADING</td><td>NOPAGE</td><td>VERT</td></tr><tr><td>DET.SUP</td><td>ID.ONLY</td><td>ONLY</td><td></td></tr></table> These keywords modify the report format. For information about them and their synonyms, see Chapter 2, “UniVerse Keywords.” If you do not specify any report qualifiers, SUM produces a report with: ■ UniVerse default heading and footing ■ Column headings on every page ■ Paged output to the screen	COL.HDR.SUPP	FIRST	ID.SUP	SAMPLE	COL.SPCS	FOOTING	LPTR	SAMPLED	COL.SUP	HDR.SUP	MARGIN	SUPP	COUNT.SUP	HEADING	NOPAGE	VERT	DET.SUP	ID.ONLY	ONLY	
COL.HDR.SUPP	FIRST	ID.SUP	SAMPLE																		
COL.SPCS	FOOTING	LPTR	SAMPLED																		
COL.SUP	HDR.SUP	MARGIN	SUPP																		
COUNT.SUP	HEADING	NOPAGE	VERT																		
DET.SUP	ID.ONLY	ONLY																			

SUM Parameters (Continued)

SUSPEND.FILES

Use SUSPEND.FILES to suspend UniVerse processes that make changes to files, without terminating user processes. You must be a UniVerse Administrator logged in to the UV account to use SUSPEND.FILES.

Syntax

SUSPEND.FILES [ON | OFF]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
ON	Database suspension is required.
OFF	Database suspension is not required.

SUSPEND.FILES Parameters

Description

SUSPEND.FILES with no options displays the current state of database suspension.

When you specify SUSPEND.FILES ON, all file modifications made in the UniVerse environment are suspended. These modifications include:

- Creating, deleting, resizing, and clearing files
- Reconfiguring files (splits, merges, extensions, renaming, etc.)
- Adding and deleting part files to a distributed file
- Creating, building, and deleting secondary indexes
- Creating, modifying, and deleting schemas
- Any changes that affect file headers
- Committing, writing, and deleting data in files
- Cataloging and decataloging programs



- Compiling I-descriptors and BASIC programs
- Updating sequential files and select lists

***Note:** External processes in progress will complete, but no new processes will be allowed to start. This means that database modifications such as transactions will finish before their processes are suspended.*

SUSPEND.RECOVERY

Use SUSPEND.RECOVERY to suspend the transaction logging system if it is currently enabled. You must be a UniVerse Administrator logged in to the UV account to use SUSPEND.RECOVERY.

Syntax

SUSPEND.RECOVERY [INFORM]

Parameter

The following table describes the parameter of the syntax:

Parameter	Description
INFORM	Displays messages during the suspend process.

SUSPEND.RECOVERY Parameter

Description

SUSPEND.RECOVERY suspends the transaction logging system. In this state, updates to recoverable files are prohibited, except by means of the roll-forward utility, and any attempt to update a recoverable file waits until the state changes. You can reenable transaction logging with [ENABLE.RECOVERY](#).

While transaction logging is in the suspended state, programs that request writes to the log file wait until the state changes.

Example

This example suspends transaction logging:

```
>SUSPEND.RECOVERY
Request to Suspended Logging Subsystem made at 12:47:59 on 01 OCT
1996. You can use the 'Display logging state' menu to verify the
current state of the logging subsystem.
```

T.ATT

Use T.ATT to assign exclusive control of a tape drive to your account. Use this command before you try to access a tape drive.

Syntax

T.ATT [MTU [*mtu*]] [[BLK] *size* | (*size*)] [MAP *mapname*] [-WAIT]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
MTU	Specifies a tape drive. Use MTU if you have more than one drive assigned to you or if the assigned drive is not drive 0.
<i>mtu</i>	Indicates the mode (<i>m</i>), number of tracks (<i>t</i>), and tape unit number (<i>u</i>) of a tape drive. T.ATT uses only the unit specification and ignores the mode and track specifications. See the MTU keyword in Chapter 2, “UniVerse Keywords.”

T.ATT Parameters

Parameter	Description
<i>size</i>	Specifies the tape block size as <i>size</i>. Subsequent tape operations use the block size specified by T.ATT. If you do not specify the block size, the default block size defined in the &DEVICE& file is used. If the block size is not defined in the &DEVICE& file, <i>size</i> is taken from field 5 of the T.ATT VOC entry. In IDEAL and INFORMATION flavor accounts, there is no default block size; variable-length tape records are written and read, if allowed by the drive. In PICK, REALITY, and IN2 flavor accounts, the default block size is 8192.
<i>mapname</i>	Specifies the name of the map you want to set for the tape device. <i>mapname</i> must have been built and installed in shared memory. You can also specify <i>mapname</i> as NONE, UTF8, or DEFAULT. NONE specifies the UniVerse internal character set. UTF8 specifies the UniVerse internal character set with the system delimiters mapped to the Unicode Private Use Area. DEFAULT specifies <i>mapname</i> as the name in the NLSDEFDEVMAP parameter in the <i>uvconfig</i> file.
-WAIT	Tells the processor to queue this assignment if the device is currently assigned to another user. After the other user unassigns the device, it is assigned to you.

T.ATT Parameters (Continued)

Description

If the tape drive is already assigned to another user, T.ATT displays an error message. Wait until the other user logs out or uses [T.DET](#) before you issue another T.ATT.

The device must have the proper permissions for you to attach it.

If the device has a map name in its definition in the [&DEVICE&](#) file, this definition is used unless the MAP keyword is specified. T.ATT with the MAP keyword overrides any map name specified in the [&DEVICE&](#) file for the tape drive until either you unassign the device, or you execute another ASSIGN or T.ATT command specifying a map name. For more information about mapping, see the *UniVerse NLS Guide*.

T.BCK

Use T.BCK to back up a specified number of tape records or to back up over a previous end-of-file mark.

Syntax

T.BCK [MTU [*mtu*]] [*nn*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
MTU	Specifies a tape drive. Use MTU if you have more than one drive assigned to you or if the assigned drive is not drive 0.
<i>mtu</i>	Indicates the mode (<i>m</i>), number of tracks (<i>t</i>), and tape unit number (<i>u</i>) of a tape drive. T.BCK uses only the unit specification and ignores the mode and track specification. See the MTU keyword in Chapter 2, “UniVerse Keywords.”
<i>nn</i>	The number of tape records you want to back up over. If you do not specify <i>nn</i> , T.BCK backs up over the previous end-of-file mark.

T.BCK Parameters

Description

If the drive encounters an end-of-file mark before it backs up over the specified number of tape records, or if you do not specify *nn*, the tape is positioned before the end-of-file mark. A subsequent read operation gets an immediate end-of-file indicator.

T.DET

Use T.DET to relinquish exclusive control of the tape drive.

Syntax

T.DET [MTU [*mtu*]]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
MTU	Specifies a tape drive. Use MTU if you have more than one drive assigned to you or if the assigned drive is not drive 0.
<i>mtu</i>	Indicates the mode (<i>m</i>), number of tracks (<i>t</i>), and tape unit number (<i>u</i>) of a tape drive. T.DET uses only the unit specification and ignores the mode and track specifications. See the MTU keyword in Chapter 2, “ UniVerse Keywords .”

T.DET Parameters

Description

The T.DET command runs a BASIC program that executes an [UNASSIGN](#) command.

T.DUMP

Use T.DUMP to copy files from disk to tape. Before you use T.DUMP, assign the tape drive using [ASSIGN](#) or [T.ATT](#).

Syntax

T.DUMP [**DICT**] *filename* [*records*] [**FROM** *n*] [*selection*] [*sort*]
[**MTU** [*mtu*]] [**BLK size**] [*modifiers*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	The file dictionary.
<i>filename</i>	The name of the file containing the records you want to dump to tape. You can specify <i>filename</i> anywhere in the sentence. T.DUMP uses the first word in the sentence that has a file descriptor in the VOC file and the file name.
<i>records</i>	Specifies the records to include in the list. Enclose record IDs in single quotation marks to prevent RetrieVe from interpreting them as field names. You can specify as many record IDs as you want, separated by spaces. T.DUMP also accepts record IDs from an active select list.
FROM <i>n</i>	Specifies that select list <i>n</i> is to be used.
<i>selection</i>	A selection expression specifying the conditions that data in a record must meet for the record to be copied to tape. A selection expression begins with the keyword WITH . For syntax details, see the WITH keyword.
<i>sort</i>	A sort expression that specifies the type of sort and the field on which to sort. The syntax is as follows: BY <i>field</i> Sort on <i>field</i> in ascending order. BY.DSND <i>field</i> Sort on <i>field</i> in descending order.

T.DUMP Parameters

Parameter	Description
	BY.EXP <i>field</i> Explode multivalues in <i>field</i> and <i>sort</i> in ascending order. BY.EXP.DSND <i>field</i> Explode multivalues in <i>field</i> and <i>sort</i> in descending order. For more information about sort expressions, see Chapter 2, “UniVerse Keywords.” When NLS locales are enabled, T.DUMP uses the Collate convention of the current locale to determine the collating order for any existing sort expression. If an index has its own Collate convention defined, T.DUMP uses it instead of the current locale definition. For more information, see the <i>UniVerse NLS Guide</i>.
MTU	Specifies a tape drive. Use MTU if you have more than one drive assigned to you, or if the assigned drive is not drive 0.
<i>mtu</i>	Indicates the mode (<i>m</i>), number of tracks (<i>t</i>), and tape unit number (<i>u</i>) of a tape drive. You can use the default for these values, 000, which indicate ASCII mode, 9-track tape, and unit number 0. The track specification is otherwise ignored. See the MTU keyword in Chapter 2, “ UniVerse Keywords .”
BLK <i>size</i>	Indicates the block size if the block size is different from the default (exact size depends on the tape device you are using).

T.DUMP Parameters (Continued)

modifiers can be any of the following:

Modifier	Description
HEADER <i>text</i>	Specifies <i>text</i> to include in the tape label. Note that REALITY format text is 16 characters maximum. PICK format text is 47 characters maximum, but this 47-character maximum must also include the filename and a space before the text.
HDR.SUP	Suppresses the tape label.

T.DUMP Modifiers

Modifier	Description
PICK.FORMAT	Specifies Pick tape format.
INFORMATION.FORMAT	Specifies Prime INFORMATION tape format.
REALITY.FORMAT	Specifies REALITY tape format.

T.DUMP Modifiers (Continued)

Description

The simplest form of the T.DUMP command is as follows:

T.DUMP *filename*

This syntax writes all records in the data file to tape. If you want to save records in the data file and the file dictionary, execute T.DUMP separately for the data file and the dictionary. T.DUMP puts an end-of-file mark at the end of each operation.

At the end of a series of consecutive T.DUMPs, two additional end-of-file marks are written. This occurs when you switch from write mode to read mode.

If you specify INFORMATION.FORMAT, T.DUMP ignores HEADER and HDR.SUP.

Note: Multireel tape handling is supported only for device types DC, DT, and F in PICK flavor accounts.



T.EOD

Use T.EOD to advance a tape to the end-of-data mark. The end-of-data mark is two consecutive end-of-file marks.

Syntax

T.EOD [MTU [*mtu*]]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
MTU	Specifies a tape drive. Use MTU if you have more than one drive assigned to you or if the assigned drive is not drive 0.
<i>mtu</i>	Indicates the mode (<i>m</i>), number of tracks (<i>t</i>), and tape unit number (<i>u</i>) of a tape drive. T.EOD uses only the unit specification and ignores the mode and track specifications. See the MTU keyword in Chapter 2, “UniVerse Keywords.”

T.EOD Parameters

T.FWD

Use T.FWD to advance a specified number of tape records or to the end-of-file mark.

Syntax

T.FWD [MTU [*mtu*]] [*nn*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
MTU	Specifies a tape drive. Use MTU if you have more than one drive assigned to you or if the assigned drive is not drive 0.
<i>mtu</i>	Indicates the mode (<i>m</i>), number of tracks (<i>t</i>), and tape unit number (<i>u</i>) of a tape drive. T.FWD uses only the unit specification and ignores the mode and track specifications. See the MTU keyword in Chapter 2, “ UniVerse Keywords .”
<i>nn</i>	The number of tape records to advance.

T.FWD Parameters

Description

If the drive encounters an end-of-file mark before it advances over the specified number of tape records, or if you do not specify *nn*, the tape is positioned after the next end-of-file mark.

T.LOAD

Use T.LOAD to copy files created by [T.DUMP](#) from tape to disk. Use [ASSIGN](#) to assign the tape drive before you use T.LOAD.

Syntax

T.LOAD [**DICT**] *filename* [*selection*] [**MTU** [*mtu*]] [*modifiers*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
<i>filename</i>	The name of the file into which you want to load the records. You can specify <i>filename</i> anywhere in the sentence. T.LOAD uses the first word in the sentence that has a file descriptor in the VOC file as the file name.
<i>selection</i>	A selection expression specifying the conditions that data in a record must meet for the record to be loaded from tape to disk. A selection expression begins with the keyword WITH . For syntax details, see the WITH keyword in Chapter 2, “ UniVerse Keywords .”
MTU	Specifies a tape drive. Use MTU if you have more than one drive assigned to you or if the assigned drive is not drive 0.
<i>mtu</i>	Indicates the mode (<i>m</i>), number of tracks (<i>t</i>), and tape unit number (<i>u</i>) of a tape drive. T.LOAD uses only the unit specification and ignores the track and mode specification. See the MTU keyword in Chapter 2, “ UniVerse Keywords .”

T.LOAD Parameters

modifiers can be any of the following:

Modifier	Description
OVERWRITING	Overwrites records that already exist on disk. If you do not include this keyword, T.LOAD loads only those records that do not already exist.
PICK.FORMAT	Specifies Pick tape format.
INFORMATION.FORMAT	Specifies Prime INFORMATION tape format.
REALITY.FORMAT	Specifies REALITY tape format.

T.LOAD Modifiers



Note: *If the configurable parameter THDR512 is 0, the tape label size for a PICK or REALITY flavor cartridge tape created in UniVerse differs from the size for one created on a Pick system. (The length of the tape label on a Pick system is always 512.) If you want to copy a file that was created on a Pick system, assign the tape with a block size of 512, then use T.LOAD to copy the file to disk. If you want to copy a file that was created on a UniVerse system in a PICK or REALITY flavor account, use T.READ to determine the block size, reassign the tape to this block size, and reposition the tape before using T.LOAD to copy the file to disk.*

If THDR512 is set, UniVerse always reads and writes tape labels with a size of 512 characters regardless of tape block size.

Multiple tape handling is supported only for device types DC, DT, and F in PICK and REALITY flavor accounts.

T.RDLBL

Use T.RDLBL to read a tape label at the beginning of a reel.

Syntax

T.RDLBL [MTU [*mtu*]] [*n*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
MTU	Specifies a tape drive. Use MTU if you have more than one drive assigned to you or if the assigned drive is not drive 0.
<i>mtu</i>	Indicates the mode (<i>m</i>), number of tracks (<i>t</i>), and tape unit number (<i>u</i>) of a tape drive. The default for these values, 000, indicates ASCII mode, 9-track tape, and unit number 0. T.RDLBL ignores the track specification. See the MTU keyword in Chapter 2, “UniVerse Keywords.”
<i>n</i>	Specifies a tape reel number. If it does not match the actual tape reel number, T.RDLBL displays the label with an error message.

T.RDLBL Parameters

Description

T.RDLBL reads the tape label and displays the reel number, block size, time, and date from when the tape was written, and a header text. T.RDLBL changes tape block size if necessary and prints a message that it has done so. If the tape does not have a label, the message `Tape was unlabelled` appears.



Note: If the tape is not positioned at the beginning of a reel, T.RDLBL does not try to read the tape label, returning to the UniVerse system prompt. If the tape is positioned at the beginning of a logical tape file, it tries to read the label, but if T.RDLBL cannot find the label, it displays a message like `Tape was unlabelled`.

T.READ

Use T.READ to read a tape record and then display it.

Syntax

T.READ [MTU [*mtu*]] [*options*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
MTU	Specifies a tape drive. Use MTU if you have more than one drive assigned to you or if the assigned drive is not drive 0.
<i>mtu</i>	Indicates the mode (<i>m</i>), number of tracks (<i>t</i>), and tape unit number (<i>u</i>) of a tape drive. The default for these values, 000, indicates ASCII mode, 9-track tape, and unit number 0. T.READ ignores the track specification. See the MTU keyword in Chapter 2, “UniVerse Keywords.”

T.READ Parameters

options can be any of the following:

Option	Description
LPTR	Prints tape records on the system printer.
HEX	Lists data in hexadecimal as well as character format.
FROM <i>n</i>	Reads the tape beginning at tape record <i>n</i> relative to the current position of the tape.
TO <i>n</i>	Stops reading after tape record <i>n</i> .

T.READ Options

Description

If you manually rewind the tape, a subsequent T.READ does not reposition the tape to the beginning. It reads the tape starting at the position of the previous T.READ because UniVerse maintains its own internal tape position counters.

T.REW

Use T.REW to rewind a tape to the load point.

Syntax

T.REW [MTU [*mtu*]]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
MTU	Specifies a tape drive. Use MTU if you have more than one drive assigned to you or if the assigned drive is not drive 0.
<i>mtu</i>	Indicates the mode (<i>m</i>), number of tracks (<i>t</i>), and tape unit number (<i>u</i>) of a tape drive. The default for these values, 000, indicates ASCII mode, 9-track tape, and unit number 0. T.REW ignores the track specification. See the MTU keyword in Chapter 2, “UniVerse Keywords.”

T.REW Parameters

T.SPACE

Use T.SPACE to advance a specified number of files on the tape or to advance to the end-of-file mark. If the drive encounters an end-of-file mark before it advances over the specified number of records, the tape is positioned after the next end-of-file mark.

Syntax

T.SPACE [MTU [*mtu*]] [*n*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
MTU	Specifies a tape drive. Use MTU if you have more than one drive assigned to you or if the assigned drive is not drive 0.
<i>mtu</i>	Indicates the mode (<i>m</i>), number of tracks (<i>t</i>), and tape unit number (<i>u</i>) of a tape drive. T.SPACE uses only the unit specification and ignores the mode and track specifications. See the MTU keyword in Chapter 2, “UniVerse Keywords.”
<i>n</i>	The number of files to space forward on the tape. If you do not specify <i>n</i> , T.SPACE displays a prompt requesting the number of files.

T.SPACE Parameters

T.UNLOAD

Use T.UNLOAD to rewind a magnetic tape and unload the tape from the drive after you finish using it.

Syntax

T.UNLOAD [MTU [*mtu*]]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
MTU	Specifies a tape drive. Use MTU if you have more than one drive assigned to you or if the assigned drive is not drive 0.
<i>mtu</i>	Indicates the mode (<i>m</i>), number of tracks (<i>t</i>), and tape unit number (<i>u</i>) of a tape drive. The default for these values, 000, indicates ASCII mode, 9-track tape, and unit number 0. T.UNLOAD ignores the track specification. See the MTU keyword in Chapter 2, “UniVerse Keywords.”

T.UNLOAD Parameters

T.WEOF

Use T.WEOF to write an end-of-file mark on the tape at the current position.

Syntax

T.WEOF [MTU [*mtu*]]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
MTU	Specifies a tape drive. Use MTU if you have more than one drive assigned to you or if the assigned drive is not drive 0.
<i>mtu</i>	Indicates the mode (<i>m</i>), number of tracks (<i>t</i>), and tape unit number (<i>u</i>) of a tape drive. The default for these values, 000, indicates ASCII mode, 9-track tape, and unit number 0. T.WEOF ignores the track specification. See the MTU keyword in Chapter 2, “ UniVerse Keywords .”

T.WEOF Parameters

T.WTLBL

Use T.WTLBL to write a tape label in a format compatible with a Pick or REALITY system. The label is written at the current position of the tape. The label includes the time and date.

Syntax

T.WTLBL [MTU [*mtu*]] [*format*] [*text*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description						
MTU	Specifies a tape drive. Use MTU if you have more than one drive assigned to you or if the assigned drive is not drive 0.						
<i>mtu</i>	Indicates the mode (<i>m</i>), number of tracks (<i>t</i>), and tape unit number (<i>u</i>) of a tape drive. The default for these values, 000, indicates ASCII mode, 9-track tape, and unit number 0. T.WTLBL ignores the track specification. See the MTU keyword in Chapter 2, “ UniVerse Keywords .”						
<i>format</i>	Specifies the tape label format. It can be one of the following: <table><tr><td>PICK.FORMAT</td><td>The Pick tape format.</td></tr><tr><td>INFORMATION.FORMAT</td><td>The Prime INFORMATION tape format. It is the same as the Pick tape format.</td></tr><tr><td>REALITY.FORMAT</td><td>The Microdata REALITY tape format.</td></tr></table>	PICK.FORMAT	The Pick tape format.	INFORMATION.FORMAT	The Prime INFORMATION tape format. It is the same as the Pick tape format.	REALITY.FORMAT	The Microdata REALITY tape format.
PICK.FORMAT	The Pick tape format.						
INFORMATION.FORMAT	The Prime INFORMATION tape format. It is the same as the Pick tape format.						
REALITY.FORMAT	The Microdata REALITY tape format.						
<i>text</i>	Specifies text to include in the tape label. <i>text</i> must be a single word. Do not enclose <i>text</i> in quotation marks. To include more than one word in text, separate the words with periods.						

T.WTLBL Parameters

Description

Reel number is 01. Multireel tape handling has not yet been implemented.

Pick labels (including ADDS Mentor and Ultimate) are 80 bytes long and have the following format (p represents an ASCII blank, **␣** represents an item mark, and **F** represents a field mark):

␣ L p xxxx p HH:MM:SS
p DD p MMM p YYYY
p filename p header pad **F** rr

PICK.FORMAT

␣	Item mark (segment mark)
L	ASCII character
xxxx	Tape record size (hexadecimal)
HH:MM:SS	Time (external 24-hour)
DD MMM YYYY	Date (external form)
filename	Name of the file dumped
header	(Optional) Label header text
pad	Enough blanks to get to byte 78
F	Field mark (attribute mark)
rr	Reel number (hexadecimal)

REALITY labels are 78 bytes long and have the following format (p represents an ASCII blank, **␣** represents an item mark, **F** represents a field mark, and **v** represents a value mark):

␣ L header **v** HH:MM:SS
p DD p MMM p YYYY
F rr **F** xxxxx **F** **␣** pad

REALITY.FORMAT

I	Item mark (segment mark)
L	ASCII character L
<i>header</i>	(Optional) Label header text (<= 44 characters)
V	Value mark (follows L if no header)
<i>HH:MM:SS</i>	Time (external 24-hour)
<i>DD MMM YYYY</i>	Date (external form)
F	Field mark (attribute mark)
<i>rr</i>	Reel number (decimal)
<i>xxxxx</i>	Tape record size (decimal)
<i>pad</i>	Fill blanks up to 78 bytes

NLS Mode

T.WTLBL maps the label text to the external character set of the tape. This text is never truncated. For more information about character sets, see the *UniVerse NLS Guide*.

Example

The following example specifies text to include in a tape label:

>T.WTLBL SALES.JANUARY



TANDEM

Use TANDEM to display or control the input and output displayed on another user's terminal from your terminal. You must be a UniVerse Administrator logged in to the UV account to use TANDEM.

Note: This command is not supported on Windows platforms.

Syntax

TANDEM *terminal.no*

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>terminal.no</i>	The terminal number of the terminal whose activity you want to watch or control.

TANDEM Parameter

Description

TANDEM shows output only from UniVerse processes. You can use TANDEM in the following three modes:

Mode	Description
Feed	Lets you enter commands for another user on your terminal. If you enter data at the same time as the other user, your keystrokes are defined by the system implementation of the terminal driver.
Message	Lets you send text to another user's terminal. You cannot control the location or format of the characters displayed on the user's terminal. Data is not treated as input for the other user.
View	Default mode. Shows another user's input and output on your terminal. This display continues when you use message or feed mode.

TANDEM Modes

While using TANDEM, you can change modes by entering an escape sequence (Esc followed by an action code). You can use any of the following sequences:

Sequence	Description
Esc D	Puts TANDEM in view mode. Terminates message or feed mode if active. Sends a BREAK to the other user's process.
Esc F	Puts TANDEM in feed mode. Terminates message mode if active.
Esc M	Puts TANDEM in message mode. Terminates feed mode if active.
Q	In view mode, same as Esc X. No effect in other modes.
Esc U	Same as Esc D.
Esc V	Puts TANDEM in view mode. Terminates message or feed mode if active.
Esc X	Ends the TANDEM session.
Esc Esc	In message mode or feed mode, lets you send Esc to another user's terminal.
Esc ?	Displays information on your terminal. Also displays the status information for the current session, such as the active mode.

TANDEM Escape Sequences



To exit the TANDEM process, type **Q** then press Return at your terminal. It can take a moment for the process to quit.

A user's *terminal number* is the first number in the output of a **WHO** command.

***Note:** The **COMO** command does not capture output produced by TANDEM.*

When you use TANDEM on a phantom process, you cannot use feed or message mode. To determine the number of the phantom process, use the **PORT.STATUS** command to show the pid of the job. Convert the hexadecimal value of the shared memory segment to a decimal value. Use this decimal value to watch the output of a phantom process. For example, the phantom process pid 23456 uses the shared memory segment hexadecimal value 0xaceba460. You can use the equivalent decimal value 42080 to watch the output of the phantom process.

When you finish watching the remote terminal, you cannot invoke TANDEM again until an I/O operation is performed from the remote terminal.

The feed mode for a TANDEM session is not supported on all platforms. The system displays a message in this case.

TERM

Use TERM to set or display printer and terminal characteristics. You can also use TERM to set the terminal type.

Syntax

TERM [*#1*, *#2*, *#3*, *#4*, *#5*, *#6*, *#7*, *#8*, *#9*]

Parameters

#1 through *#8* are positional parameters. Each parameter is optional, but its position must be held by a comma.

Parameter	Description
<i>#1</i>	The line length of the terminal screen. <i>#1</i> must be an integer between 11 and 32,767. The default is a page width of 79.
<i>#2</i>	The number of lines displayed on the terminal screen. The default is a page depth of 24.
<i>#3</i>	The number of lines skipped at the bottom of the screen. This number is subtracted from page depth (<i>#2</i>). The default is 0.
<i>#4</i>	A number indicating the amount of delay following a linefeed. The default is 0, or no delay. 1 is the least amount of delay; 7 is the most.
<i>#5</i>	A number indicating the amount of delay following a formfeed. The default is 2. Range is from 2 through 3.
<i>#6</i>	The ASCII number corresponding to the Backspace key. The default is 8.
<i>#7</i>	The number of characters per line on the line printer. The default is 132.
<i>#8</i>	The number of lines per page on the line printer. The default is 66.
<i>#9</i>	The map name for the terminal device or the auxiliary printer.

TERM Parameters

Description

If you use TERM with no qualifiers, TERM displays the current settings.

You can change the characteristics of print channel 0 or another print channel using SETPTR. The characteristics that you set using SETPTR override the printer characteristics that you set using TERM.

You can use [HUSH](#) and PTERM to change other characteristics of the terminal.

TERM assumes that the first nonnumeric parameter is a terminal type.

NLS Mode

Use TERM to display the names of the maps associated with the terminal and the auxiliary printer. For more information about mapping, see the *UniVerse NLS Guide*.

Examples

To set the number of characters per line, lines per screen, and lines per printer page, enter the following:

```
>TERM 40,22,,,,,33
```

To check the current settings, enter the following:

```
>TERM
Terminal Printer

Page width:      79      132
Page depth:      24      66
Page skip :      0
LF delay  :      0
FF delay  :      2
Backspace :      0
ic16404
>
```

To set terminal type to wy50 (Wyse 50), enter the following:

```
>TERM WY50
>
```

To set an auxiliary map in NLS mode, enter the following:

```
>TERM ,,,,,,,,,,MNEMONICS
```

TIME

Use TIME to display the current system time and date on your terminal. TIME uses a 24-hour clock.

Syntax

TIME

Description

NLS Mode.

TIME uses the uses the Time convention of the current locale to display information. For more information, see the *UniVerse NLS Guide*.

Example

```
>TIME
11:10:00 20 OCT 1995
>
```



UMASK

Use UMASK to set who can read, write, and execute new files you create. You can also use UMASK to display the current file creation mask.

***Note:** This command is not supported on Windows platforms.*

Syntax

UMASK [*nnn*]

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>nnn</i>	An octal number that specifies the file creation mask. If you do not specify a number, UMASK displays the current file creation mask.

UMASK Parameter

Description

UMASK values are in octal format. The default value UMASK is the default as set on your UNIX system. The default lets you read, write, and execute your files, and users can read and execute them.

Use the following list to choose the proper value for UMASK. Add the numbers for the permissions you do *not* want to grant for your files and use this number as the value of UMASK.

Octal Number	Permission
400	Owner can read file.
200	Owner can write to file.
100	Owner can execute file.
40	Group can read file.
20	Group can write to file.
10	Group can execute file.
4	Other can read file.
2	Other can write to file.
1	Other can execute file.

UMASK Values

For example, if you want to be able to read, write, and execute your files and you want the members of your group to be able to execute them, but you do not want to allow any other access to your files, UMASK should equal 1+2+4+20+40, or 67.

Examples

```
>UMASK
Current UMASK is: 2

      OWNER              GROUP              OTHER
Read Writ Exec  Read Writ Exec  Read Writ Exec
  ON   ON   ON   ON   ON   ON   ON  OFF  ON
```

To change the file creation mask, enter the following:

```
>UMASK 67
```

This is what the file creation mask looks like when UMASK=67:

```
>UMASK
```

```
Current UMASK is: 67
```

OWNER			GROUP			OTHER		
Read	Writ	Exec	Read	Writ	Exec	Read	Writ	Exec
ON	ON	ON	OFF	OFF	ON	OFF	OFF	OFF

UNASSIGN

Use UNASSIGN to release a peripheral device from your control.

Syntax

UNASSIGN *device*

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>device</i>	The device that you want to unassign. <i>device</i> must be defined in the &DEVICE& file.

UNASSIGN Parameter

Description

While the device is assigned to you, no one else can use it. UNASSIGN releases a device so that it becomes available to other users. You can unassign a device that has been assigned to your exclusive use from your terminal. If you are using UNASSIGN at your terminal, you must be at the same terminal where you assigned the device.

When you log out or quit and you have devices that are still assigned to you, UniVerse automatically unassigns them.

For more information, see [&DEVICE&](#) in Chapter 4, “[Local and System Files.](#)”

UNICODE.FILE

Use UNICODE.FILE in NLS mode to convert an unmapped file to a mapped file, or to convert a mapped file to an unmapped file. You cannot use UNICODE.FILE to convert a B-tree file (type 25) or a distributed file.

Syntax

UNICODE.FILE [**DICT**] *filename* [*mapname*] [**TEST** [**ID.ONLY**]]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Converts records in the file dictionary of <i>filename</i> . If you do not specify DICT, converts records in the data file.
<i>filename</i>	The name of the file you want to convert. If <i>filename</i> is unmapped, it is converted to a mapped file with the specified <i>mapname</i> . If <i>filename</i> is mapped, it is converted to an unmapped file with a map name of NONE.
<i>mapname</i>	The name of the map the file uses for conversion from internal to external byte sequences. <i>mapname</i> must be built and installed in shared memory. Do not specify <i>mapname</i> when converting from external to internal byte sequences; the map currently associated with the file is used. If you are converting an unmapped file to a mapped file, you must specify <i>mapname</i>. To use the default map file settings of NLSDEF-FILEMAP and NLSDEFDIRMAP in the <i>uvconfig</i> file, specify <i>mapname</i> as DEFAULT. You cannot specify <i>mapname</i> if the file already has a map name other than NONE assigned to it.
TEST	Verifies that no data loss occurs during the file conversion. If data is lost, the file is not converted and a report is displayed.
ID.ONLY	Checks only the record IDs for data loss.

UNICODE.FILE Parameters

Description

The conversion process requires exclusive access to the file.

If any characters cannot be converted using the resulting output map, the records are instead written to the **&UNICODE.FILE&** file, a dynamic file in the local directory with a map of NONE. The record IDs of these records are stored in a saved select list named BAD.IDS.

If you use the TEST keyword, no records are written to the &UNICODE.FILE&, but the record IDs of any records the process cannot convert are saved in the BAD.IDS select list.

@SYSTEM.RETURN.CODE returns a value of 0 if the command succeeds, or a value less than 0 if there is an error.

Example

This example converts the mapped ACCOUNTS file to an unmapped file:

```
>UNICODE.FILE ACCOUNTS
Checking that file's data can be read using KSC5601 NLS map.
100 records read.

Converting file's data to UNICODE.
100 records converted.
```

UNLOCK

Use UNLOCK to clear file, group, and update record locks. You must be a UniVerse Administrator logged in to the UV account to execute UNLOCK.

Syntax

```
UNLOCK [NODE node] [USER user.number]  
[FILE pathname | [DEVICE device.number] [INODE i.node.number]]  
[GROUP group.address] [RECORD record.ID]  
{ FILELOCK | GROUPLock | READULOCK | READLLOCK | ALL }
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
NODE <i>node</i>	Restricts lock removal to the specified node. UNIX. This must be the network node number, which is the last part of the TCP/IP host number specified in the <i>/etc/hosts</i> file. Windows Platforms. If valid for Windows platforms, the path of the <i>hosts</i> file is <i>/Windows/System32/Drivers/etc/hosts</i> where Windows is the Windows installation directory. On Windows platforms, you can also use the LAN Manager node name.
USER <i>user.number</i>	Restricts lock removal to locks owned by the specified user. You can get the user number from the output of LIST.READU .
FILE <i>pathname</i>	Restricts lock removal to a file or device specified by the path. You cannot specify DEVICE or INODE clauses in an UNLOCK statement that uses the FILE clause.
DEVICE <i>device.number</i>	Restricts lock removal to a file or a device specified by the device number.

UNLOCK Parameters

Parameter	Description
INODE <i>i.node.number</i>	Restricts lock removal to a file or a device specified by the i-node number.
GROUP <i>group.address</i>	Restricts lock removal to the group specified by <i>group.address</i> . The group address appears in hexadecimal format on the LIST.READU report.
RECORD <i>record.ID</i>	Restricts lock removal to the record specified by the record ID.

UNLOCK Parameters (Continued)

Specify the lock type as one of the following:

Lock Type	Description
FILELOCK	Removes only file locks.
GROUPLOCK	Removes only group locks and update record locks associated with the group.
READULOCK	Removes only update record locks.
READLLOCK	Removes only shared record locks.
ALL	Removes all locks.

UNLOCK Lock Types

Description

You can specify ID numbers (that is, the node number, device number, or record ID) in either decimal or hexadecimal format. Precede the number with 0x to specify it as hexadecimal.

UNLOCK SEMAPHORE

Use UNLOCK SEMAPHORE to unlock system semaphores. You must be a UniVerse Administrator logged in to the UV account to use the command UNLOCK SEMAPHORE.

Syntax

UNLOCK { FILELOCK | GROUPLock | READULock } SEMAPHORE *n*

UNLOCK { TLOGLOCK | T30LOCK | PSTATLOCK | LOGINLOCK } SEMAPHORE

When you use the first syntax, you must specify the semaphore type and the semaphore number.

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>n</i>	The semaphore number associated with a lock. The number can be from 1 through the total number of semaphores on the system. The number of semaphores on the system appears in the <i>uvconfig</i> file. The FSEMNUM is the number of file lock semaphores, and the GSEMNUM is the number of group lock semaphores. The semaphore number appears in the Lmode column of the LIST.READU report.
FILELOCK	Releases the specified concurrency control semaphores that control access to the file lock table.
GROUPLock READULock	Releases the specified concurrency control semaphores that control access to the group lock table. The semaphore types GROUPLock and READULock are the same because both are controlled by the group lock table. Shared record locks are also in the group lock table.
TLOGLOCK	Releases the transaction logger semaphore.

UNLOCK Parameters

Parameter	Description
T30LOCK	Releases the semaphore that gives access to dynamic file information in the shared memory segment.
PSTATLOCK	Releases the PORT.STATUS semaphore.
LOGINLOCK	Releases the login semaphore.
UNLOCK Parameters (Continued)	

Description

UNLOCK SEMAPHORE is not available on all systems. It is designed only for systems whose semaphores are implemented in assembly language.

You can specify the semaphore number in either decimal or hexadecimal format. Precede the number with 0x to specify it as a hexadecimal.

UPDATE.ACCOUNT

Use UPDATE.ACCOUNT to update the contents of your VOC file. UPDATE.ACCOUNT replaces records in your VOC file with records in the system's [NEWACC](#) file.

Syntax

UPDATE.ACCOUNT [PIOPEN | PICK | NEWACC | IN2 | INFORMATION | REALITY] [INFORM][NOPAGE | NO.PAGE]

Parameters

The following table describes each parameter of the syntax:

Parameter	Description
Account Flavor	The flavor of the UniVerse account.
INFORM	Does not display the keys that were changed, just the number of keys changed.
NOPAGE or NO.PAGE	Suppresses page breaks.

UPDATA.ACCOUNT Parameters

Description

The optional arguments PIOPEN, PICK, NEWACC, IN2, INFORMATION, and REALITY let you change the flavor of your account.

When the UniVerse administrator creates a new UniVerse account, UniVerse uses information in the NEWACC file to build the account's VOC file. As you use the account to create files, sentences, paragraphs, and other items, your VOC file changes. None of the items you create in your account appear in the NEWACC file.

However, when the UniVerse software changes (for example, when a new release is installed), your VOC file needs to be updated for records that should be the same in your VOC and in NEWACC.

When you first log on to your UniVerse account after a new release has been installed, UniVerse tells you that your VOC file is not current and asks if you want to update the account. You can copy items from the NEWACC file to your VOC file by entering **Y**. UniVerse does not delete any VOC entries that you have added.

When you modify the VOC file, you might create a record with the same name as a new record in the NEWACC file. When UniVerse updates the account, it uses the file **&TEMP&** to save any record in your VOC file that differs from a record with the same name in NEWACC. You can inspect and retrieve these records after you update your account.

Use the **VVOC** command to find out if your account is up-to-date.

UPDATE.INDEX

Use UPDATE.INDEX to update the secondary indexes of a file, delayed by the [DISABLE.INDEX](#) command.

Syntax

UPDATE.INDEX [DICT] [*filenames*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
<i>filenames</i>	The names of UniVerse files whose indexes you want to update. If you do not specify <i>filenames</i> , UPDATE.INDEX prompts for one.

UPDATE.INDEX Parameters

Description

You can use UPDATE.INDEX when automatic updating is enabled or disabled. Automatic updating is initially enabled. After disabling it, you can use [ENABLE.INDEX](#) to reenable automatic indexing. Note that enabling automatic updating does not update the index entries; use UPDATE.INDEX for this.

Reports generated from files while automatic updating is disabled can contain incorrect information. Records added cannot be accessed using a secondary key; records deleted can still be accessed using a secondary key; and records modified can be associated with the wrong secondary key.

Once automatic updating is disabled, do the following to ensure that the indexes are completely up-to-date:

1. Reenable automatic updating using [ENABLE.INDEX](#).

2. Close all opened versions of the file systemwide, even if they are reopened immediately. This step ensures that no users are still operating with automatic updating disabled.
3. Issue UPDATE.INDEX to bring the indexes up-to-date.

UPDATE.INDEX does not enable automatic updating.

Use [LIST.INDEX](#) to display the updating status of an index.



usa

Use *usa* for general spooler administration. Use *usa* from a UNIX shell. You must be a UniVerse Administrator to use all options except *-A*, *-B*, *-H*, *-j*, *-m*, *-n*, *-p*, *-s*, *-S*, and *-u*. Members of printer groups can also use the *-F*, *-a*, *±o*, and *±q* options. If you specify no options, *-s* is assumed.

Note: This command is not supported on Windows NT systems.

Syntax

```
usa [ p printer [ F[a[d] formlist [ a [ +number | pathname ] ] ] ]
[ A [ b [ B [ c [ C [ g [ H [ j [ ±L [ m [ n [ ±o [ P [ ±q
[ r [ R [ s [ S [ u [ v [ z ]
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
-p	If you do not specify the <i>-p</i> option, <i>usa</i> assumes all printers.
-F[a[d]	Mounts one or more forms on <i>printer</i> . <i>formlist</i> is a list of form names separated by commas (no spaces before or after the comma). Each form name can be up to 32 characters long. If one or more forms are currently mounted, <i>-F</i> replaces them with the new form list. <i>-Fa</i> adds the forms in <i>formlist</i> to the list of currently mounted forms. <i>-Fd</i> deletes the forms in <i>formlist</i> from the list of currently mounted forms. You must be a UniVerse Administrator or a member of a printer group to use <i>-F</i> .
-a	Used with the <i>-F</i> and <i>-p</i> options, <i>-a</i> verifies the alignment of <i>printer</i> . If the argument to <i>-a</i> is <i>+number</i> , <i>usa</i> uses the <i>number</i> of lines of the first waiting job of that form. If the argument is <i>pathname</i> , <i>usa</i> uses data in the file <i>pathname</i> for verification. <i>pathname</i> must be the file's absolute UNIX path. You must be a UniVerse Administrator or a member of a printer group to use <i>-a</i> .

usa Parameters

Parameter	Description
	After printing an alignment test, <i>usa</i> asks if you want to repeat the test. To reprint the alignment test, enter n at the prompt.
-A	Lists the status of spooler queues, displaying only active print jobs.
-b	Stops printing the current job (does not kill).
-B	Lists the status of the BOGUS spooler queue.
-c	Restarts printing of the current job. You must be a UniVerse Administrator to use -c.
-C	Resets the job ID number to 0 (zero). You must be a UniVerse Administrator to use -C.
-g	Defines a printer group. You must be a UniVerse Administrator to use -g.
-H	Displays a short help message.
-j	Lists queued print jobs.
±L	Disables (-) or enables (+) spooler logging to the <i>err.log</i> and <i>act.log</i> files. You must be a UniVerse Administrator to use ±L.
-m <i>form</i>	Lists the status of all print jobs on printers using <i>form</i> .
-n <i>port</i>	Lists the status of all print jobs on port. <i>port</i> is a user's port number.
±o	Disables (-) or enables (+) printing. You must be a UniVerse Administrator or a member of a printer group to use ±o.
-P	Suppresses automatic pagination. You must be a UniVerse Administrator to use -P.
±q	Disables (-) or enables (+) queuing. You must be a UniVerse Administrator or a member of a printer group to use ±q.
-r	Resets printer configuration. You must be a UniVerse Administrator to use -r.

usa Parameters (Continued)

Parameter	Description
-R	Simulates a crash restart by resetting the spooler daemon and rereading the <i>sp.config</i> and <i>usplog</i> files. After reading <i>usplog</i>, <i>usa</i> restores information about the spool queue to its precrash state. If changes were made to <i>sp.config</i>, <i>usa</i> deletes information about the spooler queue. You must be a UniVerse Administrator to use -R.
-s	Lists the status of all spooler queues.
-S	Lists the status of spooler queues, displaying only the queues containing print jobs (active and inactive).
-u <i>user</i>	Lists the status of all print jobs of <i>user</i> . <i>user</i> is a valid UNIX user ID.
-v	Lists all mounted forms.
-z	Shuts down the spooler. You must be a UniVerse Administrator to use -z.

usa Parameters (Continued)



usd

Use *usd* to start up the spooler daemon. Use *usd* from a UNIX shell. The *usd* command requires UniVerse Administrator privileges.

Note: This command is not supported on Windows NT systems.

Syntax

```
usd [directory] [-a filename] [-b] [-e filename] [-L] [-r seconds] [-t]  
[-w seconds]
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>directory</i>	The name of the spooler directory. If you do not specify <i>directory</i> , the daemon uses <i>usr/spool/uv</i> as the spooler directory.
-a	Creates a new spooler activity logging file with the name <i>filename</i> . <i>filename</i> must be the full path of the activity logging file.
-b	Enables BOGUS printer functionality.
-e	Creates a new spooler error logging file with the name <i>filename.filename</i> . <i>filename</i> must be the full path of the error logging file.
-L	Creates default activity and error logging files <i>err.log</i> and <i>act.log</i> in the spooler directory (<i>/usr/spool/uv</i>).

usd Parameters

Parameter	Description
-r	Specifies the number of seconds the spooler should wait after encountering an error condition before retrying the <i>usd</i> command. This allows printing to continue, for example, after you reload paper in the printer. The default value is 60 seconds.
-t	Specifies that jobs be processed in first-in, first-out order.
-w	Specifies the number of seconds the spooler should wait on UNIX system calls such as open and device setup calls. If the operating system does not return from these calls, the spooler waits for the specified number of seconds before it disables the printer and goes on to the next printer. The default value is 10 seconds.

usd Parameters (Continued)

Description

If you specify no options, *usd* creates no log files, and jobs are processed in order by size (smaller jobs are processed before larger).

The spooler daemon uses two files during startup, *sp.config* and *usplog*. The *sp.config* file contains printer definitions. The *usplog* file contains the image of the last queue state, which is used to rebuild the queue after a crash. At startup time the spooler daemon looks for these files in the default spooler directory (usually */usr/spool/uv*). If you specify *directory* in the *usd* command at startup time, the spooler daemon looks for *sp.config* and *usplog* in the specified directory, and the *usp* command spools print files to that directory.

You can spool files from within UniVerse to a directory other than */usr/spool/uv*.

USERS

Use USERS to display the number of users currently logged in to the system.

Syntax

USERS

Example

```
>USERS  
There are currently 7 users logged on the system.  
>
```



usm

Use *usm* to alter the specification of a job already in the printer queues. Use *usm* from a UNIX shell.

Note: *This command is not supported on Windows NT systems.*

Syntax

```
usm [-B [L] string [-E [L] string [-F formname] [-h] [-H] [-k]
      [-n copies] [-p printer] [-P priority] [-q] [-r] [-t delay]
      [-x n [-m]] [-y n [-m]] jobnum | all
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
-B	Begins printing from the first character of <i>string</i> . If <i>string</i> contains blanks, enclose the entire string in quotation marks.
-BL	Begins printing at the line containing <i>string</i> . If <i>string</i> contains blanks, enclose the entire string in quotation marks.
-E	Ends printing at the last character of <i>string</i> . If <i>string</i> contains blanks, enclose the entire string in quotation marks.
-EL	Ends printing at the line containing <i>string</i> . If <i>string</i> contains blanks, enclose the entire string in quotation marks.
-F	Changes the form to <i>formname</i> .
-h	Changes the job status to HOLD (that is, make the print file a hold file). Holds the job only until releasing it for printing.
-H	Displays a short help message.
-k	Kills a print job.
-n	Changes the number of copies to <i>copies</i> .

usm Parameters

Parameter	Description																
-p	Moves a print job to the queue for printer <i>printer</i> .																
-P	Changes priority to <i>priority</i> .																
-q	Retains the print job as a hold file after printing. Retains the job as a hold file even after releasing it for subsequent printing.																
-r	Prints a hold file.																
-t	Changes the relative or absolute time <i>delay</i> when to print. You can specify delays in the following formats:																
	<table> <tr> <th>Syntax</th><th>Example</th></tr> <tr> <td><i>+minutes</i></td><td>+5</td></tr> <tr> <td><i>+hours:minutes</i></td><td>+2:15</td></tr> <tr> <td><i>+days.hours:minutes</i></td><td>+7.00:00</td></tr> <tr> <td><i>hour:minute</i></td><td>12:00</td></tr> <tr> <td><i>day[hour:minute]</i></td><td>15.12:00</td></tr> <tr> <td><i>month.day[.hour:minute]</i></td><td>6.15.12:00</td></tr> <tr> <td><i>year.month.day[.hour:minute]</i></td><td></td></tr> </table>	Syntax	Example	<i>+minutes</i>	+5	<i>+hours:minutes</i>	+2:15	<i>+days.hours:minutes</i>	+7.00:00	<i>hour:minute</i>	12:00	<i>day[hour:minute]</i>	15.12:00	<i>month.day[.hour:minute]</i>	6.15.12:00	<i>year.month.day[.hour:minute]</i>	
Syntax	Example																
<i>+minutes</i>	+5																
<i>+hours:minutes</i>	+2:15																
<i>+days.hours:minutes</i>	+7.00:00																
<i>hour:minute</i>	12:00																
<i>day[hour:minute]</i>	15.12:00																
<i>month.day[.hour:minute]</i>	6.15.12:00																
<i>year.month.day[.hour:minute]</i>																	
-x	Prints page <i>n</i> or pages <i>n-m</i> .																
-y	Prints line <i>n</i> or lines <i>n-m</i> .																
<i>jobnum</i>	Specifies the print job number.																
all	Specifies all print jobs belonging to the user.																
usm Parameters (Continued)																	

usp

Use *usp* to submit a print job to the spooler. Use *usp* from a UNIX shell.

This command is not supported on Windows NT systems.

Syntax

usp [-B [L] *string* [-C *classtext*] [-e] [-E [L] *string* [-f] [-F *formname*] [-h] [-H] [-J *jobtext*] [-l] [-n *copies*] [-p *printer*] [-P *priority*] [-q] [-Q] [-r] [-s] [-t *delay*] [-x n [-m]] [-y n [-m]]] *files*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
-B	Begins printing from the first character of <i>string</i> . If <i>string</i> contains blanks, enclose the entire string in quotation marks.
-BL	Begins printing at the line containing <i>string</i> . If <i>string</i> contains blanks, enclose the entire string in quotation marks.
-C	Prints <i>classtext</i> in banner line 3.
-e	Suppresses the end-of-job page eject.
-E	Ends printing at the last character of <i>string</i> . If <i>string</i> contains blanks, enclose the entire string in quotation marks.
-EL	Ends printing at the last character of <i>string</i> . If <i>string</i> contains blanks, enclose the entire string in quotation marks.
-f	Source file contains FORTAN forms control codes.
-F	Waits until form <i>formname</i> is mounted on the printer.
-h	Suppresses printing of the banner page.
-H	Displays a short help message.

usp Parameters

Parameter	Description														
-J	Prints <i>jobtext</i> in banner line 2.														
-l	Prints sequential line numbers.														
-n	Prints the number of copies specified by <i>copies</i> .														
-p	Routes to <i>printer</i> instead of to the default printer.														
-P	Sets <i>priority</i> other than default.														
-q	Retains the print job as a hold file after printing. Retains the job as a hold file even after releasing it for subsequent printing.														
-Q	Sets the job status to HOLD (that is, makes the print file a hold file). Holds the job only until releasing it for printing.														
-r	Removes the source file when done.														
-s	Uses a symbolic link.														
-t	Specifies a relative or absolute time delay when to print. You can specify delays in the following formats:														
	<table> <tr> <td><i>+minutes</i></td><td>+5</td></tr> <tr> <td><i>+hours:minutes</i></td><td>+2:15</td></tr> <tr> <td><i>+days.hours:minutes</i></td><td>+7.00:00</td></tr> <tr> <td><i>hour:minute</i></td><td>12:00</td></tr> <tr> <td><i>day[.hour:minute]</i></td><td>15.12:00</td></tr> <tr> <td><i>month.day[.hour : minute]</i></td><td>6.15.12:00</td></tr> <tr> <td><i>year.month.day[.hour : minute]</i></td><td>95.6.15.12:00</td></tr> </table>	<i>+minutes</i>	+5	<i>+hours:minutes</i>	+2:15	<i>+days.hours:minutes</i>	+7.00:00	<i>hour:minute</i>	12:00	<i>day[.hour:minute]</i>	15.12:00	<i>month.day[.hour : minute]</i>	6.15.12:00	<i>year.month.day[.hour : minute]</i>	95.6.15.12:00
<i>+minutes</i>	+5														
<i>+hours:minutes</i>	+2:15														
<i>+days.hours:minutes</i>	+7.00:00														
<i>hour:minute</i>	12:00														
<i>day[.hour:minute]</i>	15.12:00														
<i>month.day[.hour : minute]</i>	6.15.12:00														
<i>year.month.day[.hour : minute]</i>	95.6.15.12:00														
-x	Prints page <i>n</i> or pages <i>n-m</i> .														
-y	Prints line <i>n</i> or lines <i>n-m</i> .														
<i>files</i>	The UNIX paths to the files to be sent to the spooler. usp sends multiple files separated by spaces to the spooler as one job with continuous pagination.														

usp Parameters (Continued)

uv

Use *uv* to enter the UniVerse environment from an operating system command prompt. UniVerse Administrators can use the *-admin* option to start up and shut down UniVerse.

Syntax

uv [*-admin option*] [*-version*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
-admin	You must be a UniVerse Administrator to use <i>-admin</i> and any of its options.
<i>option</i> is one of the following:	
-c[learshm]	Clears the BASIC catalog shared memory bit, allowing users to log in to UniVerse.
-help	Lists all available <i>-admin</i> options.
-start	Starts up UniVerse.
-stop	Shuts down UniVerse.
-L[ock]	Suspends all UniVerse file I/O. Same as SUSPEND.FILES ON .
-R[eport]	Lists current state of database suspension. Same as SUSPEND.FILES with no arguments.
-U[nlock]	Resumes suspended UniVerse file I/O. Same as SUSPEND.FILES OFF .
-version	Displays the currently installed version of UniVerse.

uv Parameters

Description

When you use `uv` with no options and the directory you are working in is not set up as a UniVerse account, the following message appears:

```
This directory is not set up for UniVerse.  
Would you like to set it up (Y/N)?
```

Enter **Y** to create a UniVerse account. UniVerse creates the necessary files, and you enter the UniVerse environment.

Enter **N** to quit and reenter the operating system environment.

UV.LOGIN

Use UV.LOGIN to initialize all UniVerse account environments.

Syntax

UV.LOGIN

Description

The UV.LOGIN entry in the VOC file of the UV account defines a sequence of commands that are executed in any UniVerse account when any user logs in to the account. For example, you might include specifying default printer settings, enabling phantom notification, or changing date formats. If the UV.LOGIN entry exists, it is executed before the LOGIN entry in the VOC file of the account.

When UniVerse is first installed, the UV account's VOC file does not contain a UV.LOGIN entry. You can create or change a UV.LOGIN entry in the UV account at any time, UniVerse searches for this entry and executes it when any user invokes UniVerse and when users log in to any UniVerse account. The UV.LOGIN entry must be a sentence, a paragraph, a menu, a proc, a BASIC program, a verb, or a remote pointer to one of these.

The UV.LOGIN entry is not executed when users log to a UniVerse account by executing a LOGTO or CHDIR command.

If you do not want a phantom to execute the UV.LOGIN entry, you must add a test to the UV.LOGIN entry for @TTY = 'phantom'. For example, if a phantom executes a UV.LOGIN paragraph containing the following IF statement, the IF statement exits the UV.LOGIN paragraph:

```
IF @TTY = 'phantom' THEN GO END.OF.UV.LOGIN
```

Put commands that phantoms should not execute in the last section of the UV.LOGIN paragraph. The last line of the UV.LOGIN paragraph must be the following label:

```
END.OF.UV.LOGIN:
```


UNIX. If you want UniVerse process with a single command parameter, such as `uv "UPDATE .MASTER"`, you can choose to avoid executing the `UV.LOGIN` entry. Use the `@tty` test just described, and have the process direct standard input, standard output, and standard error to nonterminal devices. You can execute this example from a Bourne shell:

```
uv "UPDATE.MASTER" </dev/null > /dev/null 2>&1
```

Example

```
      UV.LOGIN
0001: PARagraph to be executed in all UniVerse accounts
0002: * Disable interrupt
0003: BREAK OFF
0004: * Clear screen
0005: CS
0006: * Dispay general message to all users
0007: DISPLAY MESSAGE TO ALL EMPLOYEES:
0008: DISPLAY Today at noon in the cafeteria, everyone is
0009: DISPLAY to come on down and enjoy pizza and tonic,
0010: DISPLAY generously supplied by the marketing department.
0011: * Pause for a bit
0012: SLEEP 20
0013: * Run security program that updates security files
0014: RUN SECURITY.BPS USER.STATUS
0015: DATA "LOGIN"
0016: * Set default printer settings
0017: SETPTR , , , , , FORM LEGAL, AT MARKETING NOEJECT, BRIEF
0018: * Set autologout if no activity
0019: AUTOLOGOUT 20
0020: * Enable interrupt
0021: BREAK ON
0022: * End of commands
0023: END.OF.UV.LOGIN:
```



UV.VI

Use UV.VI to invoke the UNIX editor, *vi*, using UniVerse file-naming syntax. This is useful for type 1 files whose records have names longer than 14 characters. You can also use UV.VI to invoke *vi* on records in hashed files.

Note: *This command is not supported on Windows platforms.*

Syntax

UV.VI [DICT] *filename* [*records* | *]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies that you want to edit records in the file dictionary.
<i>filename</i>	The name of an existing file.
<i>records</i>	The names of records you want to edit. Separate multiple record IDs with spaces. You can also use an active select list.
*	Specifies all records in the file.

UV.VI Parameters

Description

If you enter only UV.VI *filename*, UniVerse prompts for a record name.

If you use UV.VI after using SAVE.LIST, UV.VI locks each record in the select list.

Example

To edit the program RECEIVABLES in the file BP, enter the following:

>UV.VI BP RECEIVABLES

uvbackup (UNIX)

Use *uvbackup* from a UNIX shell to save specified files on a daily, weekly, or comprehensive basis. You can specify files on the command line, from standard input, or from a list in a file. Output from *uvbackup* goes to standard output. For access to all system files and UniVerse files, you must be a UniVerse Administrator.

Syntax

```
uvbackup { -d | -w | -f } [ -b blksize ] [ -cachedetail ] [ -cmdfil filename ]  
[ -delay buffers ] [ -l "labeltext" ] [ -limit buffers ] [ -rev7 ] [ -rev8 ] [ -rev93 ]  
[ -rev94 ] [ -rev95 ] [ -s file ] [ { -t device } ... ] [ -v | -V ]  
[ - | pathnames ]
```

Options

Specify each option separately, and precede each option with a hyphen.

Option	Description
-d	(Daily) Backs up records in specified UniVerse hashed files that have been added or changed since the last full, weekly, or daily <i>uvbackup</i> .
-w	(Weekly) Backs up records in specified UniVerse hashed files that have been added or changed since the last full or weekly <i>uvbackup</i> . Weekly backups back up any records that have already been backed up with the -d option.
-f	(Full) Backs up all specified UniVerse and operating system files.
-b	Specifies the block size in increments of 512 bytes. The default is 8192. The minimum is 512, the maximum is defined by the configurable parameter BLKMAX.
-cachedetail	Lists shared memory cache details.
-cmdfil	Specifies a command file that contains a list of file names to back up. Each filename should be on a separate line in the command file.

uvbackup Options

Option	Description
–delay	Specifies the number of buffers to use before flushing the buffer to the backup image. Generally, <i>buffers</i> is half the number of buffers specified by the <i>–limit</i> option. The default <i>buffers</i> value is 15.
–l	Specifies <i>labeltext</i> of up to 80 characters to identify the backup. You must enclose <i>labeltext</i> in quotation marks. All characters are valid. <i>uvbackup</i> writes this text in the user label portion of the Backup Image Header.
–limit	Specifies the number of shared memory buffers to use. The default value is 30 buffers, whose maximum size can be up to 128 K (set by the <i>–b</i> option). Block sizes larger than 128 K automatically decrease the number of shared memory buffers. To avoid using shared memory buffers, set <i>buffers</i> to 1.
–rev7	Produces a backup in a format suitable for restoring to UniVerse Release 7.
–rev8	Produces a backup in a format suitable for restoring to UniVerse Release 8.
–rev93	Produces a backup in a format suitable for restoring to UniVerse Release 9.3.
–rev94	Produces a backup in a format suitable for restoring to UniVerse Release 9.4.
–rev95	Produces a backup in a format suitable for restoring to UniVerse Releases 9.5.1 through 9.5.1C.
–s	Specifies a file to which screen output is captured. You must also specify the <i>–v</i> or <i>–V</i> option to use <i>–s</i> .
–t	Specifies the device to which to write backup data. <i>device</i> can be either a path or an entry in the &DEVICE& file. Use multiple <i>–t</i> options to specify up to 10 devices. <i>uvbackup</i> writes to the first device specified, then the second, and so on.
–v	Displays paths. <i>uvbackup</i> displays all paths of files as they are backed up. If there is an error, a message describing the problem appears.
–V	Displays paths and record IDs. <i>uvbackup</i> displays all paths of files and, for UniVerse hashed files, record IDs as they are backed up. If there is an error, a message describes the problem.
–	Reads paths from standard input.
<i>pathnames</i>	List of files to back up. Each file name must be the path of the file.

uvbackup Options (Continued)

Description

uvbackup backs up files specified by *pathnames*. This means you must explicitly specify each data file, file dictionary, and secondary index you want to back up. Specify the paths of file dictionaries in the format *D_filename*. Specify the paths of secondary indexes in the format *I_filename/**.

Examples

The first three examples show how to use the *find* command to specify the files you want to back up.

This example backs up all directories and files on the system:

```
$ find / -print | uvbackup -f -v -l "FULL SYSTEM BACKUP" - > /dev/rmt/0
```

The slash (/) specifies the root directory and all its dependencies. The *-f* option specifies a full backup. The *-v* option lists paths of directories and files as they are backed up. The *-l* option specifies tape label text. The *-* option takes input from standard input. The files are backed up on to the tape drive attached to the device file */dev/rmt/0*.

The next example backs up records that have changed in all UniVerse hashed files in the */usr/work* directory and its dependencies:

```
$ find /usr/work -print | uvbackup -d -v -b 512 -t MT0
```

The *-b* option sets a block size of 512 bytes. The *-t* option specifies MT0 (defined in the *&DEVICE&* file) as the device to write to.

The next example backs up records that have changed in all UniVerse hashed files in the current directory and its dependencies:

```
$ find . -print | uvbackup -w -V - > /dev/rmt/0
```

The *-V* option lists record IDs as well as paths of directories and files as they are backed up. The *-* option takes input from standard input.

The next example shows the *-cachedetail* option output:

```
Backup Date      : Tue Jul  7 11:08:07 1998
Reel Number     : 1
Image Type      : Full Backup (ver7 UNIX)
Block Size      : 8192 bytes
NLS on          : False
Label           : CacheDetail test
```

Cache Shared Memory State:
WRITE pid=9042 READER pid=9043
Control =0 Flags =1 Delay = 14
CacheVal =536870912 CacheState=1
Current device(s): /temp/cachedet.image
Backing up /accounts/Baktests/FILEHUGE

Waiting for READER process (9043) to terminate.

Removing Cache Shared Memory Segment (272)

READER engine terminating

Total files: 1 Total bytes : 15880669 Elapsed Time: 00:00:35
0 operating system files processed, 0 broken, totalling 0 data
bytes.
1 UniVerse files processed, 0 corrupted.
38936 UniVerse records processed, 0 corrupted, totalling
15880669 data bytes.

EndOfUvbackup

uvbackup (Windows Platforms)

Use *uvbackup* from an MS-DOS window to save specified files on a daily, weekly, or comprehensive basis. You can specify files on the command line or from a list in a file. Output from *uvbackup* goes to the specified output device. For access to all system files and UniVerse files, you must be a UniVerse Administrator.

Syntax

```
uvbackup { -d | -w | -f } [ -b blksize ] [ -cachedetail ] [ -cmdfil filename ]  
[ -delay buffers ] [ -l "labeltext" ] [ -limit buffers ] [ -rev7 ] [ -rev8 ] [ -rev93 ] [ -  
rev94 ] [ -rev95 ] [ -s file ] [ { -t device } ... ] [ -v | -V ] [ pathnames ] -walk  
pathname ]
```

Options

Specify each option separately, and precede each option with a hyphen.

Option	Description
-d	(Daily) Backs up records in specified UniVerse hashed files that have been added or changed since the last full, weekly, or daily <i>uvbackup</i> .
-w	(Weekly) Backs up records in specified UniVerse hashed files that have been added or changed since the last full or weekly <i>uvbackup</i> . Weekly backups back up any records that have already been backed up with the -d option.
-f	(Full) Backs up all specified UniVerse and operating system files.
-b	Specifies the block size in increments of 512 bytes. The default is 8192. The minimum is 512, the maximum is defined by the configurable parameter BLKMAX.
-cachedetail	Lists shared memory cache details.
-cmdfil	Specifies a command file that contains a list of file names to back up. Each file name should be on a separate line in the command file.

uvbackup Options

Option	Description
-delay	Specifies the number of buffers to use before flushing the buffer to the backup image. Generally, <i>buffers</i> is half the number of buffers specified by the -limit option. The default <i>buffers</i> value is 15.
-l	Specifies <i>labeltext</i> of up to 80 characters to identify the backup. You must enclose <i>labeltext</i> in quotation marks. All characters are valid. <i>uvbackup</i> writes this text in the user label portion of the Backup Image Header.
-limit	Specifies the number of shared memory buffers to use. The default value is 30 buffers, whose maximum size can be up to 128 K (set by the -b option). Block sizes larger than 128 K automatically decrease the number of shared memory buffers. To avoid using shared memory buffers, set <i>buffers</i> to 1.
-rev7	Produces a backup in a format suitable for restoring to UniVerse Release 7.
-rev8	Produces a backup in a format suitable for restoring to UniVerse Release 8.
-rev93	Produces a backup in a format suitable for restoring to UniVerse Release 9.3.
-rev94	Produces a backup in a format suitable for restoring to UniVerse Release 9.4.
-rev95	Produces a backup in a format suitable for restoring to UniVerse Releases 9.5.1 through 9.5.1C.
-s	Specifies a file to which screen output is captured. You must also specify the -v or -V option to use -s.
-t	Specifies that the output is directed to the named tape device. <i>device</i> can be either a path or an entry in the &DEVICE& file, for example, MT0.
-v	Displays paths. <i>uvbackup</i> displays all paths of files as they are backed up. If there is an error, a message describing the problem appears.

uvbackup Options (Continued)

Option	Description
-V	Displays paths and record IDs. <i>uvbackup</i> displays all paths of files and, for UniVerse hashed files, record IDs as they are backed up. If there is an error, a message describes the problem.
-walk	Generates a list of paths of all files to back up in the current directory, including all files in all subdirectories of the current directory.
<i>pathnames</i>	The paths of the files to back up.

uvbackup Options (Continued)

Description

uvbackup backs up files specified on the command line or in a command file. You must explicitly specify each data file, file dictionary, and secondary index you want to back up. Paths of file dictionaries have the form *D_filename*. Paths of secondary indexes have the form *I_filename*.

Examples

This example backs up all the files contained in a command file called *Filelist*:

```
$ uvbackup -f -v -l "FULL SYSTEM BACKUP" -cmdfil Filelist -t MT0
```

The next example backs up the two files specified on the command line. The *-b* option specifies a block size of 512 bytes.

```
$ uvbackup -b 512 -f -v -t MT0 "d:\usr\work\file1" "d:\usr\work\file2"
```

uvdlockd

Use *uvdlockd* to administer the deadlock daemon. You must be aUniVerse Administrator to use *uvdlockd*.

Syntax

uvdlockd { [-t *time*] [-r *resolution*] [-l *location*] } | [-query] | [-stop]
| [-v *victim*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
-t <i>time</i>	The time interval (in seconds) between the deadlock daemon's successive checks of the lock-waiter tables. The default is 60 seconds.
-r <i>resolution</i>	The resolution strategy the deadlock daemon uses. It can be one of the following: 0 Selects a transaction at random. This is the default. 1 Selects the newest transaction. 2 Selects the transaction with the fewest number of locks held.
-l <i>location</i>	The location of the deadlock log file. The default is <i>uvhome/uvdlockd.log</i> .
-query	Generates a report based on a one-shot analysis of the lock-waiter tables and any detected deadlocks.
-stop	Shuts down the deadlock daemon.
-v <i>victim</i>	Specifies which user number to select as the process to abort.

uvdlockd Parameters

Description

You can use the `uvdlockd` command to automatically identify and resolve deadlocks as they occur, or you can manually fix a deadlock by selecting and aborting one of the deadlocked user processes.

Resolving Deadlocks Automatically

The deadlock daemon automatically resolves deadlocks by creating and updating a set of lockwaiter tables, which represent the state of the locking and transactional system. These tables are continually examined for evidence of a deadlock. Once the daemon detects a deadlock, it selects one of the currently involved transactions to abort, removing the deadlock.

The deadlock daemon notifies the selected transaction that a deadlock has occurred and aborts the current execution layer. This rolls back any active transactional statements and cleans up any remaining locks.

Resolving Deadlocks Manually

You can resolve deadlocks manually using the `-query` option to check for deadlocks, then using the `-v` option to select a deadlock user process to abort.

Examples

This example starts the deadlock daemon, setting it to check for deadlocks every two minutes and defining the automatic resolution strategy to select the newest transaction to abort. The default deadlock log file is used.

```
$ uvdlockd -t 120 -r 1
```

The next example uses the `-query` option to generate a report of the lock-waiter tables and the detected deadlocks:

```
$ uvdlockd -query
User 49 is waiting for
user 33 who is waiting for
user 49.
```

The next example uses the `-v` option to select user process 49 to abort:

```
$ uvdlockd -v 49
```

uvfixfile

Use *uvfixfile* to verify the integrity of UniVerse hashed files, report file statistics, and repair broken files. Use *uvfixfile* from an operating system command prompt. You must be a UniVerse Administrator to repair files.

Syntax

```
uvfixfile -f[ ile] pathname [-b[rief]] [-case setting] [-dec] [-fix]
[-help] [-hex] [-i] [-l[og] logfile] [-m[ap]]
[-o[utput] outfile] [-rev] [-s[tats]] [-t[race] groups]
[-v[erbose] level] [-z[ero] groups] [-zgroup groups]
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
-f [ile]	Specifies relative or absolute path of the file you want to process.
-b [rief]	Displays brief error messages.
-case <i>setting</i>	Sets the case for interactive mode. <i>setting</i> can be invert or noinvert.
-dec	Sets the default base to decimal.
-fix	Fixes the specified file.
-help	Displays a list of available options.
-hex	Sets the default base to hexadecimal.
-i	Uses interactive mode. See Interactive Mode for a list of commands you can use in interactive mode.
-l [og]	Dumps errors to <i>logfile</i> instead of to standard error. If you do not specify <i>logfile</i> , <i>uvfixfile</i> creates a file called uvfixlog in the current directory.
-m [ap]	Maps the primary buffer and overflow buffer layout.

uvfixfile Parameters

Parameter	Description
-o [output]	Records all output, including errors, to <i>outfile</i> . If you do not specify <i>outfile</i> , <i>uvfixfile</i> creates a file called <i>uvfixout</i> in the current directory.
-rev	Displays the <i>uvfixfile</i> revision number.
-s [tats]	Displays file statistics.
-t [race]	Limits <i>groups</i> processed. <i>groups</i> can be one of the following: <i>n</i> Specifies one group. <i>n-m</i> Specifies a range of groups. <i>n+</i> Specifies a range of groups starting with <i>n</i> and ending at the end of the file. all Specifies all groups. If you specify the <i>-i</i> options along with the <i>-t</i> option, <i>-t</i> is ignored.
-v [level]	Sets the verbosity level of error messages. <i>level</i> can be a number from 0 through 9. See Verbosity Levels for a list of levels.
-z [ero]	Resets one or more group buffers. <i>groups</i> are the same as the options listed under the <i>-t</i> option.
-zgroup	Same as <i>-z</i> .

uvfixfile Parameters (Continued)

Description

uvfixfile traces through a file's groups, identifies problem groups, and reports their location to standard error. If you specify the *-fix* option, *uvfixfile* repairs most damaged files. However, *uvfixfile* cannot detect certain file breaks, and therefore it cannot fix them.

To invoke *uvfixfile* in the UniVerse environment, use the [UVFIXFILE](#) command.

Verbosity Levels

You can set the verbosity level of error messages from the command line, in interactive mode, and in step mode. *level* is a number from 0 through 9.

Level	Display
0	Errors only
1	Diagnostic messages and errors
2	Diagnostic messages, errors, and number of group buffer being processed
3	Diagnostic messages, errors, number of group buffer being processed, and group statistics
4	Diagnostic messages, errors, number of group buffer being processed, group statistics, and record header information, including the address of the current record block, forward link, backward link, and flagword (displayed in hexadecimal format)
5	Diagnostic messages, errors, number of group buffer being processed, group statistics, record header information, and meaning of bits set in each flagword
6	Diagnostic messages, errors, number of group buffer being processed, group statistics, record header information, meaning of bits set in each flagword, and record ID
7	Diagnostic messages, errors, number of group buffer being processed, group statistics, record header information, meaning of bits set in each flagword, record ID, and data record
8	Record ID only
9	Record ID and data record only

Verbosity Levels

Interactive Mode

When you use the *-i* (interactive) option, the interactive mode prompt (:) appears. You can use the following commands in interactive mode:

Command	Description
<i>!command</i>	Runs operating system <i>command</i> .
<i>?</i>	Displays available interactive mode commands with a brief description of each.
<i>akpath</i> [<i>pathname</i>]	Changes the path of the file's secondary indexes (alternate keys) to <i>pathname</i> . If you do not specify <i>pathname</i> , displays the current path of the file's secondary indexes. To set the secondary indexes' path to an empty string, specify <i>pathname</i> as NULL.
<i>case setting</i>	Sets input case. <i>setting</i> can be <i>invert</i> or <i>noinvert</i> .
<i>ex</i> [<i>it</i>]	Exits <i>uvfixfile</i> (same as <i>quit</i>).
<i>file pathname</i>	Opens a file specified by <i>pathname</i> .
<i>free</i>	Resets the freechain pointer.
<i>gl</i>	Sets group locks while tracing.
<i>group.lock</i>	Same as <i>gl</i> .
<i>hash record.ID</i>	Displays the group to which <i>record.ID</i> hashes.
<i>help</i>	Displays available interactive mode commands with a brief description of each.
<i>locks</i>	Displays if group locks will be set.
<i>m</i> [<i>ap</i>] <i>n</i>	Maps the primary buffer and overflow buffer layout.
<i>n</i> [<i>ext</i>]	Gets the next file in a select list.
<i>nl</i>	Does not set group locks while tracing.
<i>no.lock</i>	Same as <i>nl</i> .
<i>open pathname</i>	Opens a file specified by <i>pathname</i> .
<i>q</i> [<i>uit</i>]	Exits <i>uvfixfile</i> (same as <i>exit</i>).

Interactive Mode Commands

Command	Description
rev[ision]	Displays the <i>uvfixfile</i> revision number.
set { dec hex }	Sets the default base to decimal (dec) or hexadecimal (hex).
stats	Displays file statistics.
s[tep] group#	Steps through one group buffer one record block at a time. See Step Command for a list of commands you can use when stepping through a group.
t[race] groups	Traces through one or more groups. <i>groups</i> can be the number of one group, a range of groups, a starting group number followed by a + (continues to the end of the file), or all.
uvfile filename	Opens a new file using the VOC entry.
v[level] [n]	Sets the verbosity level. <i>n</i> is a number from 0 through 9. If you do not specify <i>n</i> , <i>uvfixfile</i> displays the current verbosity level. See Verbosity Levels for a list of levels.
z[ero] groups	Resets one or more group buffers. <i>groups</i> can be the number of one group, a range of groups, a starting group number followed by a + (continues to the end of the file), or all.
zgroup groups	Same as z.

Interactive Mode Commands (Continued)

Step Command

You can use the step command only when you are in interactive mode. When you enter the step command, the step prompt (`step>`) appears. You can enter the following commands at the step prompt:

Command	Description
?	Displays available step commands.
Return	Displays the next record block.
c[ont]	Switches to trace mode and displays the rest of the group.
f[wd]	Displays the next record block.

Step Command

Command	Description
h[elp]	Displays available step commands.
q[uit]	Quits, and redisplay the interactive mode prompt (:).
set <i>parameter</i>	Sets various record block parameters. See Set Command for a list of parameters you can set.
v[level] <i>n</i>	Sets the verbosity level to <i>n</i> . <i>n</i> is a number from 0 through 9. See Verbosity Levels for a list of levels.

Step Command (Continued)

Set Command

You can use the set command only when you are in step mode. You can set any of the following parameters:

Parameter	Description
bckupflg	Sets the backup flag.
blink [<i>value</i>]	Changes the backward link.
dec	Changes default base to decimal.
flagword [<i>value</i>]	Changes the flagword.
flink [<i>value</i>]	Changes the forward link.
freeflag	Marks the record block as free.
grpresiz	Sets the relocated group bit.
hex	Changes default base to hexadecimal.
ovflwbuf	Sets the oversize buffer bit.
ovflwchn	Sets the oversize change bit.
padinrec	Sets the record block padding flag.
t30first	Sets the dynamic file t30first bit.

Set Command Parameters

Parameter	Description
t30lastb	Sets the dynamic file t30lastb bit.
transflg	(Obsolete) Sets transflg.
usenewpd	Sets the new style item padding flag.
Set Command Parameters (Continued)	

UVFIXFILE

Use UVFIXFILE to verify the integrity of UniVerse hashed files, report file statistics, and repair broken files. You must be a UniVerse Administrator to repair files.

Syntax

```
UVFIXFILE [[ DICT ] filename | [ PATH ] pathname ] [ BRIEF ] [ CASE setting ]
[ DECIMAL ] [ FIX ] [ HELP ] [ HEXADECIMAL ] [ I ] [ LOG logfile ] [ MAP ]
[ OUTPUT outfile ] [ REVISION ] [ STATS ] [ TRACE groups ] [ VLEVEL ] level ]
[ { ZERO | ZGROUP } groups ]
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary of <i>filename</i> .
<i>filename</i>	The name of a UniVerse file as defined in the VOC file. <i>filename</i> must be the first option on the command line. If a select list is active, you need not specify <i>filename</i> .
PATH	Specifies the relative or full path of the file you want to process. The PATH option must be the first option on the command line. If a select list is active, you need not use the PATH option. PATHNAME is a synonym for PATH.
BRIEF	Displays brief error messages. You can also use B to specify this option.
CASE <i>setting</i>	Sets the case for interactive mode. <i>setting</i> can be INVERT or NOINVERT.
DECIMAL	Sets the default base to decimal. DEC is a synonym for DECIMAL.
FIX	Fixes the specified file. You must be a UniVerse Administrator to repair files.
HELP	Displays a list of available options.

UVFIXFILE Parameters

Parameter	Description
HEXADECIMAL	Sets the default base to hexadecimal. HEX is a synonym for HEXADECIMAL.
I	Uses interactive mode. See Interactive Mode for a list of commands you can use in interactive mode.
LOG	Dumps errors to <i>logfile</i> instead of to standard error.
MAP	Maps the primary buffer and overflow buffer layout. You can also use M to specify this option.
OUTPUT	Records all output, including errors, to outfile.
REVISION	Displays the uvfixfile revision number. REV is a synonym for REVISION.
STATS	Displays file statistics. You can also use S to specify this option.
TRACE	Limits <i>groups</i> processes. <i>groups</i> can be one of the following: <i>n</i> Specifies one group. <i>n-m</i> Specifies a range of groups. <i>n+</i> Specifies a range of groups starting with <i>n</i> and ending at the end of the file. all Specifies all groups. If you specify the I option along with the TRACE option, TRACE is ignored. You can also use T to specify this option.
VLEVEL	Sets the verbosity level of error messages. level is a number from 0 through 9. See Verbosity Levels for a list of levels. You can also use V to specify this option.
ZERO	Resets one or more group buffers. groups are the same as those listed in the TRACE option. You can also use Z to specify this option.
ZGROUP	Same as ZERO.

UVFIXFILE Parameters (Continued)

Description

UVFIXFILE traces through a file's groups, identifies problem groups, and reports their location to standard error. If you specify the FIX option, UVFIXFILE repairs most damaged files. However, UVFIXFILE cannot detect certain file breaks, and therefore it cannot fix them.

The LOG and OUTPUT keywords create files in the current account directory. You can access these files from the current UniVerse account by referring to them as records in the type 1 file [&UFD&](#). Use the following syntax with any UniVerse command that accesses type 1 files:

```
command &UFD& { logfile | outfile }
```

If you specify LOG or OUTPUT and logfile or outfile already exists, UVFIXFILE asks if you want to use the existing file. If you enter **Y** at the prompt, UVFIXFILE appends output to the file. If you answer **N**, you return to the system prompt (>).

To invoke UVFIXFILE from an operating system command prompt, use the [uvfixfile](#) command.

Verbosity Levels

You can set the verbosity level of error messages from the command line, in interactive mode, and in step mode. *level* is a number from 0 through 9.

Level	Display
0	Errors only
1	Diagnostic messages and errors
2	Diagnostic messages, errors, and number of group buffer being processed
3	Diagnostic messages, errors, number of group buffer being processed, and group statistics
4	Diagnostic messages, errors, number of group buffer being processed, group statistics, and record header information, including the address of the current record block, forward link, backward link, and flagword (displayed in hexadecimal format)
5	Diagnostic messages, errors, number of group buffer being processed, group statistics, record header information, and meaning of bits set in each flagword

Verbosity Levels

Level	Display
6	Diagnostic messages, errors, number of group buffer being processed, group statistics, record header information, meaning of bits set in each flagword, and record ID
7	Diagnostic messages, errors, number of group buffer being processed, group statistics, record header information, meaning of bits set in each flagword, record ID, and data record
8	Record ID only
9	Record ID and data record only

Verbosity Levels (Continued)

Interactive Mode

When you use the I (interactive) option, the interactive mode prompt (:) appears. You can use the following commands in interactive mode:

Command	Description
<i>!command</i>	Runs operating system <i>command</i> .
?	Displays available interactive mode commands with a brief description of each.
akpath [<i>pathname</i>]	Changes the path of the file's secondary indexes (alternate keys) to <i>pathname</i> . If you do not specify <i>pathname</i> , displays the current path of the file's secondary indexes. To set the secondary indexes' path to an empty string, specify <i>pathname</i> as NULL.
case <i>setting</i>	Sets input case. <i>setting</i> can be invert or noinvert.
ex[<i>it</i>]	Exits UVFIXFILE (same as quit).
file <i>pathname</i>	Opens a file specified by <i>pathname</i> .
free	Resets the freechain pointer.
gl	Sets group locks while tracing.
group.lock	Same as gl.
hash <i>record.ID</i>	Displays the group to which <i>record.ID</i> hashes.

Interactive Mode Commands

Command	Description
help	Displays available interactive mode commands with a brief description of each.
locks	Displays if group locks will be set.
m[ap] <i>n</i>	Maps primary buffer and overflow buffer layout.
n[ext]	Gets the next file in a select list.
nl	Does not set group locks while tracing.
no.lock	Same as nl.
open <i>pathname</i>	Opens a file specified by <i>pathname</i> .
q[uit]	Exits UVFIXFILE (same as exit).
rev[ision]	Displays the UVFIXFILE revision number.
set { dec hex }	Sets the default base to decimal (dec) or hexadecimal (hex).
stats	Displays file statistics.
s[tep] <i>group#</i>	Steps through one group buffer one record block at a time. See Step Command for a list of commands you can use when stepping through a group.
t[race] <i>groups</i>	Traces through one or more groups. <i>groups</i> can be the number of one group, a range of groups, a starting group number followed by a + (continues to the end of the file), or all.
uvfile <i>filename</i>	Opens a new file using the VOC entry.
v[level] [<i>n</i>]	Sets verbosity level. <i>n</i> is a number from 0 through 9. If you do not specify <i>n</i> , UVFIXFILE displays the current verbosity level. See Verbosity Levels on for a list of levels.
z[ero] <i>groups</i>	Resets one or more group buffers. <i>groups</i> can be the number of one group, a range of groups, a starting group number followed by a + (continues to the end of the file), or all.
zgroup <i>groups</i>	Same as z.

Interactive Mode Commands (Continued)

Step Command

You can use the step command only when you are in interactive mode. When you enter the step command, the step prompt (`step>`) appears. You can enter the following commands at the step prompt:

Command	Description
<code>?</code>	Displays available step commands.
<code>Return</code>	Displays the next record block.
<code>c[ont]</code>	Switches to trace mode and displays the rest of the group.
<code>f[wd]</code>	Displays the next record block.
<code>h[elp]</code>	Displays available step commands.
<code>q[uit]</code>	Quits, and redisplay the interactive mode prompt (:).
<code>set parameter</code>	Sets various record block parameters. See Set Command for a list of parameters you can set.
<code>v[level] n</code>	Sets verbosity level to <i>n</i> . <i>n</i> is a number from 0 through 9. See Verbosity Levels for a list of levels.

Step Commands

Set Command

You can use the set command only when you are in step mode. You can set any of the following parameters:

Parameter	Description
<code>bckupflg</code>	Sets the backup flag.
<code>blink [value]</code>	Changes the backward link.
<code>dec</code>	Changes default base to decimal.
<code>flagword [value]</code>	Changes the flagword.
<code>fink [value]</code>	Changes the forward link.
<code>freeflag</code>	Marks the record block as free.

Set Command Parameters

Parameter	Description
grpresiz	Sets the relocated group bit.
hex	Changes default base to hexadecimal.
ovflwbuf	Sets the oversize buffer bit.
ovflwchn	Sets the oversize change bit.
padinrec	Sets the record block padding flag.
t30first	Sets the dynamic file t30first bit.
t30lastb	Sets the dynamic file t30lastb bit.
transflg	(Obsolete) Sets transflg.
usenewpd	Sets the new style item padding flag.
Set Command Parameters (Continued)	

UVHELP

Use UVHELP to display a description of a UniVerse command or keyword, a conversion code, a BASIC statement or function, a UniVerse SQL statement, or a BASIC SQL Client Interface function.

Syntax

UVHELP [BASIC | CONV | PICK | SQL | BCI | *helpfile*] [*item* | *]

UVHELP [*item*] FROM *helpfile*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
BASIC	Lists help about the BASIC command. Followed by the name of a BASIC statement or function, the BASIC keyword specifies help for a BASIC statement or function.
CONV	Specifies help for conversion codes. If you do not specify <i>item</i> , CONV lists all conversion codes. If you specify a conversion code as <i>item</i> , help for that code is listed.
PICK	Specifies help for PICK, REALITY, and IN2 flavor accounts. If you do not specify <i>item</i> , PICK lists commands with PICK-flavor-specific differences. If you specify a command as <i>item</i> , help for that command is listed.
SQL	Specifies help for SQL statements. If you do not specify <i>item</i> , SQL lists all SQL statements. If you specify an SQL statement as <i>item</i> , help for that statement is listed.
BCI	Specifies help for BASIC SQL Client Interface functions. If you do not specify <i>item</i> , BCI lists all SQL Client Interface functions. If you specify a function as <i>item</i> , help for that function is listed.

UVHELP Parameters

Parameter	Description
<i>helpfile</i>	Specifies one of the system help files: BASIC.HELP BCI.HELP CONV.HELP PICK.HELP SQL.HELP SYS.HELP
<i>item</i>	The name of a UniVerse command or keyword, a BASIC statement or function, a conversion code, an SQL statement, or a UniVerse BASIC SQL Client Interface function.
*	Lists all items specified by the preceding keyword. If you do not include a keyword in the command line, * lists help items for UniVerse commands and keywords.
FROM <i>helpfile</i>	Specifies one of the system help files.

UVHELP Parameters (Continued)

uvlictool

Use *uvlictool* to report on the current state of UniVerse licensing and to clean up current licensing. You must be a UniVerse Administrator to use *uvlictool*.

Syntax

uvlictool [report_lic] [clean_lic [-a]]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
report_lic	Lists the current licensing state.
clean_lic	Cleans up the current licensing state.
-a	Recomputes license counts at the UniVerse, package, and seat level.

uvlictool Parameters

Description

uvlictool does the following:

- Lists a report on license use at both the UniVerse and the package level.
- Identifies the process that owns the license, and lists package licenses it holds.
- Identifies the remote device holding the license.
- On UNIX systems, *uvlictool* cleans up the current licenses based on shared memory segments associated with dead processes.
- On Windows platforms, *uvlictool* cleans up the current licenses based on dead entries in the process table.
- Recomputes license counts at the UniVerse, package, and seat levels.

UVPROMPT

Use UVPROMPT to change the command-line-prompt character, the select-list-active character, and the line-continue character.

Syntax

UVPROMPT [*prompt*] [*select*] [*continue*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>prompt</i>	The new command-line-prompt character. The default is > .
<i>select</i>	The new select-list-active character. The default is the same as the command-line-prompt character.
<i>continue</i>	The new line-continue character. The default is + .

UVPROMPT Parameters

UVPROMPT uses only the first character of *prompt*, *select*, and *continue* as the prompt character.

uvrestore (UNIX)

Use *uvrestore* from a UNIX shell to restore specified UniVerse files or records saved by a previous *uvbackup* procedure. You can also restore an entire system. For access to all system files and UniVerse files, you must be a UniVerse Administrator.

Syntax

```
uvrestore [-F pathname [=newpathname] [-R record [=newrecord]]]
[-X pathname] [-b blksize] [-i [b]] [-l] [-L] [-n] [-noindex]
[-noindexfix] [-p] [-P n] [-rehash] [-s file] [-startb block]
[{ -t device } ...] [-U] [-v] [-V] [-verify] [- imagepath]
```

Options

Specify each option separately, and precede each option with a hyphen.

Parameter	Description
-F	Restores only the specified file. <i>pathname</i> must match the file name saved in the image (use the <i>-i</i> option to determine what files are in the image). If you specify <i>newpathname</i> , the file originally saved in <i>pathname</i> is restored as <i>newpathname</i> . To specify more than one file, use multiple <i>-F</i> options.
-R	Restores only the specified records. <i>record</i> must match the record ID saved in the image. There can be multiple occurrences of this option. If you specify <i>newrecord</i> , the record is written to the file as <i>newrecord</i> . You can use the <i>-R</i> option only when you specify one <i>-F</i> option. Any other use terminates the restore process.
-X	Excludes <i>pathname</i> from the restoration. To specify more than one file to exclude, use multiple <i>-X</i> options.
-b	Specifies the block size in increments of 512 bytes. The default is 8192. Minimum is 512, maximum is defined by the configurable parameter BLKMAX.
-i [<i>b</i>]	Displays an index of image contents. Nothing is restored. <i>-ib</i> shows the blocks.
-l	Displays tape label information during processing.

uvrestore Options

Parameter	Description
-L	Displays the tape label only.
-n	Disables automatic creation of files. This option is valid only when used with a full backup image. Files to be restored must exist.
-noindex	Disables automatic restoration of secondary indexes with files.
-noindexfix	Disables automatic correction of secondary index <i>pathname</i> .
-p	Prompts before restoring a file or record. Possible responses are: y Restore the file or record. (You can also press ENTER to restore the file or record.) n Do not restore the file or record. d Disable prompting after the current prompt. q Quit <i>uvrestore</i> and return to the shell prompt.
-P	Pauses after <i>n</i> lines of output.
-rehash	Forces the rehashing of records as they are restored. This option is valid only when used with a full backup image. Do not use this option with the <i>-n</i> option.
-s	Specifies a file to which screen output is captured. You must also specify the <i>-v</i> or <i>-V</i> option to use <i>-s</i> .
-startb	Starts restoring from the block specified by <i>block</i> .
-t	Reads data from <i>device</i> . <i>device</i> can be either a path or an entry in the &DEVICE& file. Use multiple <i>-t</i> options to specify up to 10 devices. <i>uvrestore</i> reads from the first device specified, then the second, and so on.
-U	Overwrites disk files with the same names as those being restored. The <i>-U</i> option deletes the disk file and creates a new file with the same type, modulo, and separation as the one on the image. If you do not specify <i>-U</i> , files in an image that are older than a disk file are not restored.
-v	Displays file names and record count as the files are restored. If there is an error, a message describing the problem appears.
-V	Displays file names and, for UniVerse hashed files, record IDs as they are restored. If there is an error, a message describing the problem appears.

uvrestore Options (Continued)

Parameter	Description
<code>-verify</code>	Verifies image integrity.
<code>-</code>	Takes the name of the restore image from standard input.
<i>imagepath</i>	Specifies the path of the backup image to restore. <i>imagepath</i> can be the name of a device or the name of a file where the image is stored. If you do not specify <i>imagepath</i> , <i>uvrestore</i> prompts for it.

uvrestore Options (Continued)

Description

Secondary indexes are automatically restored with the files they index. Use the `-noindex` option to disable automatic restoration of secondary indexes. In addition, the path of the secondary index directory is automatically updated in the file header to point to the newly restored secondary index directory. Use the `-noindexfix` option to disable automatic updating of the secondary index path.

To restore the contents of a directory and all its dependencies, append a slash followed by an asterisk (`*`) to *pathname*. For example, to restore all files and subdirectories in the SALES directory, use a command like the following:

```
$ uvrestore -F '/usr/SALES/*' -l -v /dev/rmt/0
```

To restore all files beginning with UV, use a command like the following:

```
$ uvrestore -F '/usr/SALES/UV*' -l -v /dev/rmt/0
```

Enclose the file name in single quotation marks to prevent the UNIX shell from treating the asterisk as a wildcard character.

Note: *uvrestore* does not support reading from *stdin*.

Examples

This example lists paths of all directories and files on a tape in the drive attached to the device file `/dev/rmt/0`. Nothing is restored to disk.

```
$ uvrestore -i /dev/rmt/0
```



The next example copies the file ORDERS from the tape to the file ORDERS.TMP on disk. The `-p` option prompts you before restoring the file. Once the file is restored, you can use the [COPY](#) command to copy specified records from the restored file to the ORDERS file.

```
$ uvrestore -F ORDERS=ORDERS.TMP -p /dev/rmt/0
```

uvrestore (Windows Platforms)

Use *uvrestore* from an MS-DOS window to restore specified UniVerse files or records saved by a previous *uvbackup* procedure. You can also restore an entire system. For access to all system files and UniVerse files, you must be a UniVerse Administrator.

Syntax

```
uvrestore [-F pathname [= newpathname] [-R record [= newrecord]]]
[-X pathname] [-b blksize] [-i [b]] [-l] [-L] [-n] [-nodrv] [-noindex] [-
noindexfix] [-p] [-P n] [-rehash] [-s file] [-startb block] [{ -t device } ...]
[-U] [-v | -V] [-verify] [imagepath]
```

Options

Specify each option separately, and precede each option with a hyphen.

Option	Description
-F	Restores only the specified file. <i>pathname</i> must match the file name saved in the image (use the <i>-i</i> option to determine what files are in the image). If you specify <i>newpathname</i> , the file originally saved in <i>pathname</i> is restored as <i>newpathname</i> . To specify more than one file, use multiple <i>-F</i> options.
-R	Restores only the specified records. <i>record</i> must match the record ID saved in the image. There can be multiple occurrences of this option. If you specify <i>newrecord</i> , the record is written to the file as <i>newrecord</i> . You can use the <i>-R</i> option only when you specify one <i>-F</i> option. Any other use terminates the restore process.
-X	Excludes <i>pathname</i> from the restoration. To specify more than one file to exclude, use multiple <i>-X</i> options.
-b	Specifies the block size in increments of 512 bytes. The default is 8192. Minimum is 512, maximum is defined by the configurable parameter BLKMAX.
-i[b]	Displays an index of image contents. Nothing is restored. <i>-ib</i> shows the blocks.

uvrestore Options

Option	Description
-l	Displays tape label information during processing.
-L	Displays the tape label only.
-n	Disables automatic creation of files. This option is valid only when used with a full backup image. Files to be restored must exist.
-nodrv	Strips the drive letter from restored paths. This option is used to restore files onto a different disk.
-noindex	Disables automatic restoration of secondary indexes with files.
-noindexfix	Disables automatic correction of secondary index path.
-p	Prompts before restoring a file or record. Possible responses are: y Restore the file or record. (You can also press ENTER to restore the file or record.) n Do not restore the file or record. d Disable prompting after the current prompt. q Quit <i>uvrestore</i> and return to the shell prompt.
-P	Pauses after <i>n</i> lines of output.
-rehash	Forces the rehashing of records as they are restored. This option is valid only when used with a full backup image. Do not use this option with the <i>-n</i> option.
-s	Specifies a file to which screen output is captured. You must also specify the <i>-v</i> or <i>-V</i> option to use <i>s</i> .
-startb	Starts restoring from the block specified by <i>block</i> .
-t	Reads data from <i>device</i> . <i>device</i> can be either a path or an entry in the &DEVICE& file. Use multiple <i>-t</i> options to specify up to 10 devices. <i>uvrestore</i> reads from the first device specified, then the second, and so on.
-U	Overwrites disk files with the same names as those being restored. The <i>-U</i> option deletes the disk file and creates a new file with the same type, modulo, and separation as the one on the image. If you do not specify <i>-U</i> , files in an image that are older than a disk file are not restored.

uvrestore Options (Continued)

Option	Description
-v	Displays file names and record count as the files are restored. If there is an error, a message describing the problem appears.
-V	Displays file names and, for UniVerse hashed files, record IDs as they are restored. If there is an error, a message describing the problem appears.
-verify	Verifies image integrity.
<i>imagepath</i>	Specifies the path of the backup image to restore. <i>imagepath</i> is the name of a file where the image is stored. If you do not specify <i>imagepath</i> , <i>uvrestore</i> prompts for it.

uvrestore Options (Continued)

Description

Secondary indexes are automatically restored with the files they index. Use the *-noindex* option to disable automatic restoration of secondary indexes. In addition, the pathname of the secondary index directory is automatically updated in the file header to point to the newly restored secondary index directory. Use the *-noindexfix* option to disable automatic updating of the secondary index path.

To restore the contents of a directory and all its dependencies, append a backslash followed by an asterisk (*) to *pathname*. For example, to restore all files and subdirectories in the SALES directory, use a command like the following:

```
D:\IBM\UV>uvrestore -F "\usr\SALES\*" -l -v -t MT0 -nodrv
```

To restore all files beginning with UV, use a command like the following:

```
D:\IBM\UV>uvrestore -F "\usr\SALES\UV*" -l -v -t MT0 -nodrv
```

Enclose the file name in double quotation marks.

Examples

This example lists paths of all directories and files on a tape in the drive attached to the device defined as MT0 in the &DEVICE& file. Nothing is restored to disk.

```
D:\IBM\UV>uvrestore -i -t MT0
```

The next example copies the file ORDERS from the tape to the file ORDERS.TMP on disk.

```
D:\IBM\UV>uvrestore -F ORDERS=ORDERS.TMP -p -t MT0
```

The `-p` option prompts you before restoring the file. Once the file is restored, you can use the [COPY](#) command to copy specified records from the restored file to the ORDERS file.

VCATALOG

Use VCATALOG to compare the object code of a program in the system catalog space with the object code for that program in a file in your account.

Syntax

VCATALOG [*filename* [[*catalog*] [*program*]]] [LOCAL]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>filename</i>	The name of the file containing the program source code. File names can include only ASCII characters, no multibyte characters. The source code must be compiled before you use VCATALOG. UniVerse assumes that the object code is in the corresponding object code file named <i>filename.O</i> .
<i>catalog</i>	The name of the program in the system catalog space. Catalog names can include only ASCII characters, no multibyte characters.
<i>program</i>	The record in the file <i>filename.O</i> that contains the program object code. Program names can include only ASCII characters, no multibyte characters.
LOCAL	Specifies that the program being verified is a locally cataloged program.

VCATALOG Parameters

Description

This command verifies only the executable code. VCATALOG is not needed for locally cataloged programs because UniVerse does not copy the object code to the system catalog space.

If you do not specify *filename*, VCATALOG prompts you to enter the qualifier values one at a time. If you press ENTER at any of the prompts, VCATALOG terminates without verifying.

If you do not specify *catalog*, VCATALOG uses the program name as the catalog name.

If the object code in the system catalog space matches the object code in file *filename.O*, this message appears:

Program verifies.

If the object codes are different, the following message appears:

Program does not verify.

VERIFY.DF

Use VERIFY.DF to examine the headers of a distributed file and its part files, and list inconsistencies.

Syntax

VERIFY.DF *dist.filename*

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>dist.filename</i>	The name of a distributed file.

VERIFY.DF Parameter

Description

VERIFY.DF reads the distributed file header and opens each part file to verify its path and part number. It then displays any inconsistencies.

Use the [REBUILD.DF](#) command to fix distributed file inconsistencies.

Example

This example reports that PARTFILE2's part number was changed, PARTFILE4 was moved, and PARTFILE1's part number is invalid:

```
>VERIFY.DF DIST.FILE
```

```
Examining "PARTFILE2".  
Part number in PARTFILE2 has changed from 2 to 3.
```

```
Examining "PARTFILE4".  
Partfile "PARTFILE4" has been moved to /usr/ACCOUNTS/PARTFILE4.
```


Examining "PARTFILE1".
Part file "PARTFILE1" has an invalid part number of 0.
Invalid or missing Partblock for "PARTFILE1".

Examining "PARTFILE3".

VERIFY.SQL

Use VERIFY.SQL to examine the SQL catalog for inconsistencies and report any that are found. You can also use VERIFY.SQL to fix the inconsistencies.

You must be a valid SQL user to run VERIFY.SQL. You must have proper operating system permissions and SQL privileges on all tables, views, and schemas you want to verify. You also need DBA authority to use certain keywords.

Syntax

VERIFY.SQL { **TABLE** | **VIEW** } { *table* | *pathname* } [**FIX**] [*options*]

VERIFY.SQL **SCHEMA** [*schema* | *pathname*] [**FIX**] [*options*]

VERIFY.SQL { **CATALOG** | **ALL** } [**FIX**] [*options*]

VERIFY.SQL **SCHEMAS** [*options*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
TABLE	Verifies the contents of the table's SICA against data in the SQL catalog.
VIEW	Verifies the contents of the view's SICA against data in the SQL catalog.
SCHEMA	Verifies data about the schema and the SICAs of all its tables and views against data in the SQL catalog. If you do not specify the name or path of a schema, the current schema is verified.
SCHEMAS	Verifies the owners, paths, and schema names of all schemas on the system against data in the SQL catalog. When you use this keyword, the VERIFY.SQL command can take a long time to execute, because it looks at all directories on the system.
CATALOG	Verifies the internal consistency of the SQL catalog.

VERIFY.SQL Parameters

Parameter	Description
ALL	Verifies all SQL objects on the system, including the internal consistency of the SQL catalog, against data in the SQL catalog. If you do not have DBA authority, you may see many messages saying you lack SQL privileges to verify schemas and tables. When you use this keyword, the VERIFY.SQL command can take a long time to execute, because it looks at all directories on the system.
<i>table</i>	The name of an SQL table or view defined in the VOC file of the current schema.
<i>schema</i>	The name of an SQL schema.
<i>pathname</i>	The full pathname of a table, view, or schema.
FIX	Fixes all inconsistencies found. You must have DBA authority to use the FIX keyword with the ALL and CATALOG keywords.

VERIFY.SQL Parameters (Continued)

options can be any of the following:

Option	Description
LPTR	Sends output to the printer, instead of the terminal.
NOPAGE	Suppresses automatic paging of terminal output. NOPAGE is ignored if you use it in the same sentence with LPTR.
BRIEF	Suppresses terminal output. Use this option when you want to fix SQL catalog inconsistencies but do not want to display the VERIFY.SQL report on your screen. BRIEF is ignored if you use it in the same sentence with LPTR.

VERIFY.SQL Options

Description

VERIFY.SQL compares data in the security and integrity constraints areas (SICAs) of tables, views, and schemas to data in the SQL catalog, and displays any inconsistencies.

VERIFY.SQL is a diagnostic tool that you should use if you suspect information in the SQL catalog is inconsistent with the schemas, tables, views, directories, and files on your system. Such inconsistencies should not occur during normal use, but they can happen when you use operating system commands to manipulate UniVerse files (such as deleting, copying, and moving).

Users who do not have DBA authority can verify only tables, views, and schemas they have operating system permissions and SQL privileges to access. DBA administrators can verify all tables, views, and schemas on the system, provided they have proper file and directory permissions.

Fixing SQL Catalog Inconsistencies

The FIX keyword changes the data in the SQL catalog to make it agree with data in the schemas' VOC files and in the SICAs of its tables and views. If data is found in the SQL catalog for a schema, table, or view that does not exist, the data in the SQL catalog is deleted. If SQL catalog data is internally inconsistent, the data is changed to make it agree with the data found in the SQL objects it is currently verifying.

If there is no corresponding data in the SQL catalog, the inconsistencies are fixed as follows:

Command	Description
VERIFY.SQL SCHEMA <i>pathname</i> FIX	Creates the SQL catalog data using information in the schema's VOC file and in the SICAs of the schema's tables.
VERIFY.SQL SCHEMA FIX	
VERIFY.SQL TABLE <i>pathname</i> FIX	Creates the SQL catalog data using information in the SICA of the table.
VERIFY.SQL VIEW <i>pathname</i> FIX	Creates the SQL catalog data using information in the SICA of the view.

Fixes for SQL Catalog Inconsistencies



Command	Description
VERIFY.SQL SCHEMA <i>schema</i> FIX	No SQL catalog data is changed and an error message appears. VERIFY.SQL cannot locate a schema by name if the name is not in the SQL catalog.
VERIFY.SQL TABLE <i>table</i> FIX	No SQL catalog data is changed and an error message appears. VERIFY.SQL cannot locate a table by name if the name is not in the SQL catalog.
VERIFY.SQL ALL FIX	Creates the SQL catalog data using information in all the schemas' VOC files and in the SICAs of their tables. If an SQL table is not in a valid schema, no SQL catalog data is changed and an error message appears.

Fixes for SQL Catalog Inconsistencies (Continued)

Note: The *FIX* option of *VERIFY.SQL* changes the *SQL* catalog only; it does not make changes to the *VOC* file, the *SICAs* of tables, or any *UniVerse* files.

Examples

This example finds one error, because the name of the table *INV* was changed to *INVTY* (using an operating system command). The name of the dictionary was also changed, a *VOC* entry for *INVTY* was created, and the *VOC* entry for *INV* was deleted.

```
>VERIFY.SQL SCHEMA ACCOUNTS LPTR
VERIFY.SQL SCHEMA ACCOUNTS LPTR 12:54:12pm 13 Jul 1996 PAGE 1

Checking permission.
Building table list.....Done.

Verifying table '/usr/CUST'.
Checking file permissions.
Doing verify on table '/usr/CUST'.
Checking Column 'CUSTNO'.
Checking Column 'BILLTO'.
Checking Column 'PHONE'.
Checking Column 'PHONEDESC'.
Checking Association 'PHONES'.
Checking UV_USERS data for owner 'george'.

Verifying table '/usr/INVTY'.
Checking file permissions.
Possible Moved or Copied table.
Doing verify on table '/usr/INVTY'.
```

```

! The operating system table name 'INVTY' does not match table name 'INV'
  found in the SICA.
* Table has been renamed.
* SQL catalog data for table 'INV (ACCOUNTS)' should be moved.
Checking Column 'PRODNO'.
Checking Column 'DESCRIP'.
Checking Column 'QOH'.
Checking Column 'COST'.
Checking Column 'SELL'.
Checking Column 'REORDER'.
* The path in the catalog data for 'INV (ACCOUNTS)' doesn't match the path
  to the table 'INVTY'.
Checking UV_USERS data for owner 'george'.
* No SQL catalog data for table '/usr/INVTY' in the UV_USERS record for
  user 'george'.

Verifying table '/usr/ORD'.
Checking file permissions.
Doing verify on table '/usr/ORD'.
Checking Column 'ORDERNO'.
Checking Column 'DATE'.
Checking Column 'CUSTNO'.
Checking Column 'PRODNO'.
Checking Column 'QTY'.
Checking Association 'BOUGHT'.
Checking UV_USERS data for owner 'george'.

Verifying table 'INV'.
Checking file permissions.
* There is no table 'INV' in the schema 'ACCOUNTS'.
* SQL catalog data for table 'INV (ACCOUNTS)' should be deleted.

1 information-only condition found.
4 fixable errors found.

Items marked with a '!' are information messages only.
Items marked with a '*' can be fixed by using the FIX option to VERIFY.SQL.
Items marked with a '**' are situations where VERIFY.SQL could not
continue.
The next example uses the FIX keyword to rename the table INV to INVTY:
>VERIFY.SQL TABLE /usr/INVTY FIX LPTR
VERIFY.SQL TABLE /usr/INVTY FIX LPTR 12:56:28pm 13 Jul 1995 PAGE 1

Checking file permissions.
Possible Moved or Copied table.
Doing verify on table '/usr/INVTY'.
! The operating system table name 'INVTY' does not match table name 'INV'
  found in the SICA.
* Table has been renamed.
* Moving table data from table 'INV (ACCOUNTS)' to table
  '/usr/INVTY (ACCOUNTS)'.
* Moving column data from table 'INV (ACCOUNTS)' to table
  '/usr/INVTY (ACCOUNTS)'.
* Moving association data from table 'INV (ACCOUNTS)' to table
  '/usr/INVTY (ACCOUNTS)'.
* Moving ownership record for user 'george'.
Checking Column 'PRODNO'.
Checking Column 'DESCRIP'.
Checking Column 'QOH'.
Checking Column 'COST'.
Checking Column 'SELL'.

```

Checking Column 'REORDER'.
Checking UV_USERS data for owner 'george'.

1 information-only condition found.
1 error fixed.

Items marked with a '!' are information messages only.
Items marked with a '*' have been fixed.
Items marked with a '**' are situations where VERIFY.SQL could not
continue.



VI

Use VI to invoke the UNIX editor, *vi*. *vi* is a full-screen editor that can be used to create and edit BASIC programs and to edit records in a type 1 or type 19 file.

Note: *This command is not supported on Windows platforms.*

Syntax

VI [*pathname*]

Parameter

The following table describes the parameter of the syntax.

Parameter	Description
<i>pathname</i>	The UNIX path of the file you want to edit. You can use either a relative or an absolute path.

VI Parameter

Description

You cannot create a UniVerse file using VI. You must first create the file using [CREATE.FILE](#).

Type 1 and type 19 files are implemented as UNIX directories. Records in type 1 and type 19 files are UNIX files. When you use VI, specify the UNIX path for the record to edit rather than a standard UniVerse record specification. Note that the UNIX path can be different from the UniVerse file name and record name.

See your UNIX documentation for information about the *vi* editor.

If you want to use *vi* to edit a record in a hashed file, use the [UV.VI](#) command.

Example

To edit the program RECEIVABLES in the file BP, enter:

```
>VI BP/RECEIVABLES
```

VLIST

Use VLIST to display a listing of BASIC object code. VLIST displays each line of source code followed by the lines of object code it generated. VLIST also displays statistics about your program.

Syntax

VLIST [*filename*] *program* [R]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>filename</i>	The name of the file containing the source code of the BASIC program. The default <i>filename</i> is BP.
<i>program</i>	The name of the program to list.
R	Specifies to use internal reference numbers for variables and constants rather than the source code names and values.

VLIST Parameters

Example

```
>VLIST BP TO.LIST
Main Program "BP.O/TO.LIST"
Compiler version: 5.1.5
Object Level      : 4
Machine type      : 0x0
Local Variables   : 1
Subroutine args   : 0
Unnamed Common    : 0
Named Commed Seg: 0
Object Size       : 0x1e
Source lines      : 4

      1: FOR I = 1 TO 10
0001 0000 : 09a forinit          1
0001 0004 : 09e fornxt          I to 1 001c
```

```
      2: PRINT I
0002 0010 : 103 printf      I
      3: NEXT I
0003 0016 : 0c2 jump      0004:

      4: END
0005 001c : 190 stop
```

VVOC

Use VVOC to verify the contents of your VOC file by comparing your VOC file to the system's [NEWACC](#) file.

Syntax

VVOC [NO.PAGE] [LPTR]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
NO.PAGE	Suppresses automatic paging.
LPTR	Sends output to the printer.

VVOC Parameters

Description

When the UniVerse administrator creates a new UniVerse account, UniVerse uses information in the NEWACC file to build the account's VOC file. As you use the account to create files, sentences, paragraphs, and other items, your VOC file changes. None of the item you create in your account appear in the NEWACC file.

However, when the UniVerse software changes (for example, when a new release is installed), your VOC file needs to be updated for records that should be the same in your VOC and in NEWACC.

VVOC compares each record in your VOC file against the records in the NEWACC file. If VVOC finds records that do not match, it lists the differences on the terminal or printer and in one of three select lists saved in the [&SAVEDLISTS&](#) file. These saved lists are named [&VOC.ONLY](#), [&NEWACC.ONLY](#), and [&VOC.DIFF](#).

Saved List	Description
&VOC.ONLY	Contains IDs for records that are in your VOC file but not in NEWACC. All your file definitions, sentences, paragraphs, phrases, menus, and so on, are in this list.
&NEWACC.ONLY	Contains IDs for records that are in NEWACC but not in your VOC file. Under most circumstances, you should not have any records in this list. However, you should notify your system f to verify that this is so. Your UniVerse administrator may have removed some records for your installation. If so, the names of those records appear in this list.
&VOC.DIFF	Contains IDs for records that are in NEWACC and your VOC file but are different. If you have any records in this list, you may have some errors, or your installation may be unique. Check with your UniVerse administrator.

VVOC Saved Lists

To use any of these select lists, use the [GET.LIST](#) command.

VVOC only lists differences, it does not reconcile them. Use [UPDATE.ACCOUNT](#) to update your VOC file.



WALLET.ADD.KEY

Use the WALLET.ADD.KEY command to add an encryption key to an encryption wallet.

***Note:** Keys that are entered in a wallet must have a password.*

Syntax

WALLET.ADD.KEY <wallet_id> <wallet_password> <key_id> <key_password>

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
<i>wallet_id</i>	The ID of the encryption wallet.
<i>wallet_password</i>	The password to the encryption wallet.
<i>key_id</i>	The ID of the encryption key you want to add to the wallet.
<i>key_password</i>	The password of the encryption key you want to add to the encryption wallet.

WALLET.ADD.KEY Parameters

WALLET.REMOVE.KEY

Use the WALLET.REMOVE.KEY to remove an encryption key from an encryption wallet.

Syntax

WALLET.REMOVE.KEY [FORCE] <wallet_id> <wallet_password> <key_id>
[key_password]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
FORCE	If you specify FORCE, a root or uvadmin user can delete an encryption key from an encryption wallet without knowing the key password.
wallet_id	The ID of the wallet from which you want to delete an encryption key.
wallet_password	The password to the wallet from which you want to delete an encryption key.
key_id	The encryption key you want to remove from the wallet.
key_password	The password to the encryption key you want to remove from the wallet.

WALLET.REMOVE.KEY Parameters

WHO

Use WHO to display your terminal number and the name of your current account directory. If your login name is different from the account directory name, WHO also displays your login name.

Syntax

WHO

Examples

The output of the WHO command is similar to the following:

```
1 >WHO
3 mjones
>
```

This indicates that you are working in a directory named *mjones* on terminal 3.

If you used LOGTO to move to another account, the output of the WHO command is similar to the following:

```
1 >WHO
3 accounting From mjones
>
```

accounting indicates you are working in a directory named *accounting*, and *From mjones* indicates you are logged in as *mjones*.

UniVerse Keywords

Introduction	2-11
Keyword Symbols	2-11
Field Modifier Keywords	2-12
Report Qualifier Keywords	2-14
Keyword Descriptions	2-16
(...)	2-16
@ASSOC_ROW	2-16
1NF	2-16
32BIT	2-16
64BIT	2-17
A	2-17
-ACCEPT	2-17
ADDING	2-17
AFTER	2-17
ALL	2-17
ALL.MATCH	2-18
AND	2-18
ANY	2-18
APPEND	2-18
ARCHIVE	2-19
ARE	2-19
AS	2-19
-AS	2-19
ASSOC	2-19
ASSOC.WITH	2-20
AT	2-20
-AT	2-21

Be the

COL.SUP.	2-33
COLLATE	2-33
COMPLETE	2-33
CONCURRENT	2-34
CONV.	2-34
COPIES	2-34
-COPIES	2-34
COUNT.SUP.	2-34
CRT	2-35
CTYPE	2-35
DATA	2-35
DBL.SPC	2-35
DEFAULT	2-36
DEFAULTS	2-36
DEFER	2-36
DELETE	2-36
DELETING	2-37
DET-SUPP	2-37
DET.SUP	2-37
DETAIL	2-37
DEVICE	2-37
DEVICELIST	2-38
DICT	2-38
DIFF	2-38
DISABLE LOCK.HIST	2-38
DISKS.	2-38
DISPLAY.	2-38
DISPLAY.LIKE	2-39
DISPLAY.NAME	2-39
DOWN	2-39
DYNAMIC	2-39
EJECT.	2-40
EMPTY.NULL	2-40
ENABLE LOCK.HIST	2-40
ENDPAGE	2-40
ENUMERATE	2-41

EQ	2-41
EQUAL	2-41
EVAL	2-41
EVERY	2-42
EXPLODE	2-42
EXTERNAL	2-43
FILE	2-43
FILE.OFF	2-43
FILE.ON	2-43
FILELOCK.	2-43
FILEMAP	2-43
FIRST	2-44
FIX	2-44
FMT	2-44
FOOTING	2-45
FOR	2-47
FORCE	2-47
FORM	2-47
–FORM	2-47
FORM.FEED	2-47
–FORM.FEED.	2-48
FROM	2-48
FTN	2-48
FUNDAMENTAL	2-48
GE	2-49
GEN	2-49
GENERAL	2-49
GRAND-TOTAL	2-49
GRAND.TOTAL	2-49
GREATER	2-50
GROUP.SIZE	2-50
GROUPLOCK.	2-50
GT	2-51
HDR-SUPP.	2-51
HDR.SUP	2-51
–HEAD	2-51

HEADER	2-51
HEADING	2-51
–HEX	2-54
HOLD	2-54
HUSH	2-54
ID.ONLY	2-54
ID-SUP	2-54
ID-SUPP	2-55
ID.SUP	2-55
IN	2-55
IN2	2-55
IN2.FORMAT	2-55
INFO	2-55
INFORM	2-55
INFORMATION	2-56
INFORMATION.FORMAT	2-56
INODE	2-56
INPLACE	2-56
INQUIRING	2-56
INTERNAL	2-56
INTERSECT	2-57
INVERT	2-57
INVISIBLE	2-57
IS.NULL	2-57
IS.NOT.NULL	2-57
ISOLATION	2-58
JOIN.BUFFER	2-58
KEEP	2-58
KEEP.COMMON	2-58
–L	2-58
LARGE.RECORD	2-59
LE	2-59
LENGTH	2-59
LESS	2-59
LIKE	2-59
LIST	2-60

-LIST	2-60
LNUM	2-60
LOCAL	2-60
LOCK.HIST	2-60
LOCK.WAIT	2-60
LOCKS	2-61
LPTR	2-61
LT	2-61
MAP	2-61
MARGIN	2-62
MATCHES	2-62
MATCHING	2-62
MAX.	2-64
MERGE.LOAD	2-65
MFILE.HIST	2-65
MIN	2-65
MINIMIZE.SPACE	2-66
MINIMUM.MODULUS	2-67
-MODIFY	2-67
MONETARY	2-67
MTU	2-67
MULTI.VALUE	2-68
MVDISPLAY	2-68
NE	2-68
NEEDNL	2-68
NETWORK	2-69
NEW.PAGE	2-69
NEWACC	2-69
NEXT	2-69
NEXT.AVAILABLE	2-69
NFMT	2-70
NHEAD.	2-70
NO.INDEX.	2-70
NO.MATCH	2-70
NO.NEW	2-70
NO.NULLS	2-71

NO.PAGE	2-71
NO.SELECT	2-71
NO.SPLIT	2-71
NO.WAIT	2-71
NO.WARN	2-72
NODE	2-72
NODEFAULT	2-72
NOEJECT	2-72
NOFMT	2-72
NOHEAD	2-72
–NOHEAD	2-72
NOHOLD	2-73
NOKEEP	2-73
NONE	2-73
NOPAGE	2-73
NOT	2-74
NOT.MATCHING	2-74
NOXREF	2-76
NULL	2-76
NUM.SUP	2-76
ODBC.CONNECTIONS	2-76
OF	2-76
OFF	2-77
ON	2-77
ONLY	2-77
OPTIM.SCAN	2-77
OVERWRITING	2-78
PCT	2-78
PDICT	2-78
PERCENT	2-78
PERCENT.GROWTH	2-79
PERCENTAGE	2-79
PICK	2-79
PICK.FORMAT	2-79
PID	2-80
PIOPEN	2-80

PORT	2-80
–PORTNO	2-80
PREFIX	2-80
PRINT	2-81
PRINTER	2-81
PROGRAMSIZE	2-82
PROMPT	2-82
–RANGE	2-83
READLOCK.	2-83
REALITY	2-83
REALITY.FORMAT.	2-83
RECORD	2-84
RECORD.SIZE	2-84
–REJECT	2-84
REMOVING	2-85
REPORTING	2-85
REQUIRE.SELECT	2-85
RESTORE	2-86
RESTOREDATA	2-86
RETAIN.	2-86
RUNNING	2-86
SAID.	2-86
SAMPLE	2-87
SAMPLED.	2-87
SAVEDATA	2-87
SAVING	2-87
SCHEMAS.	2-88
SELECT.BUFFER	2-88
SELECT.ONLY	2-88
SEQ.NUM	2-89
SET	2-89
SHOW	2-89
SINGLE.VALUE	2-89
SOME	2-89
SPLIT.LOAD	2-90
SPOKEN	2-90

SPOOL	2-90
-SPOOL	2-90
SQL	2-90
SQUAWK	2-90
STATISTICS.	2-91
STATS.	2-91
SUPP	2-92
-SUPPRESS.	2-92
SUSPENDED	2-92
SYSTEM.	2-92
TAPE	2-92
TEMPL	2-92
TEST	2-93
THAN.	2-93
THE	2-93
THEN	2-93
TIME	2-93
TO	2-93
TOTAL	2-94
TRANSPORT	2-95
TRAP	2-95
TRUNCATE.	2-96
TTY.ON	2-96
TYPE	2-96
-UNICODE	2-96
UNION	2-97
UNIQUE	2-97
UNLIKE	2-97
UP	2-97
UPDATING	2-97
USER	2-97
USERS	2-98
USING	2-98
UTF8	2-98
VERIFIELD	2-99
VERIFILE	2-99

VERIFY. 2-99

VERT 2-100

VERTICALLY 2-100

WHEN 2-101

WITH 2-102

WITHIN 2-104

-XREF 2-105

Keywords in a UniVerse sentence let you modify the action of the command. Using keywords, you can do the following:

- Specify the value of a field in relation to other values, such as greater than, equal to, or less than.
- Sort the values in ascending or descending order.
- Verify that field values exist in another field.
- Design margins, headings, footers, and column sizes for reports.
- Summarize data and display aggregate results for reports.
- Specify breakpoints and display subtotals in a report.
- Specify a magnetic tape unit.
- Make a sentence more like colloquial English.

Introduction

The tables in the following sections list selected keywords grouped by function. The tables list keyword symbols (such as relational operators), field modifiers, field qualifiers, and report qualifiers. The rest of this chapter describes all UniVerse keywords in alphabetical order.

Keyword Symbols

The following table lists symbols you can use as keywords.

Symbol	Synonyms	Description
<	LT LESS BEFORE	Less than operator
<= =<	LE	Less than or equal to operator
=	EQ EQUAL	Equal to operator
>= =>	GE	Greater than or equal to operator
>	GT GREATER AFTER	Greater than operator
<> ×	NE NO	Not equal
#		
~	SAID SPOKEN	Sounds like

UniVerse Keywords

Symbol	Synonyms	Description
%	PCT PERCENT PERCENTAGE	Calculate percentages
&	AND	The logical operator AND
(		Open parenthesis
)		Close parenthesis
UniVerse Keywords (Continued)		

Field Modifier Keywords

The following table lists keywords you can use to modify field expressions in Retrieve sentences. For information about field expressions, see the *Guide to Retrieve*.

Keyword	Synonyms	Description
AVG	AVERAGE	Calculates and lists the average for the field.
BREAK.ON	BREAK-ON	Specifies breakpoints in a report.
BREAK.SUP		Specifies breakpoints in a report, suppressing the BREAK.ON display.
CALC	CALCULATE	Specifies TOTAL calculations. Operates only on I-descriptor fields. Specifies a calculation using the TOTAL function included in the I-descriptor field definition.
ENUMERATE	ENUM	Counts and lists the total number of values for the field.
MAX		Calculates and lists the maximum value for the field.
MIN		Calculates and lists the minimum value for the field.
Field Modifier Keywords		

Keyword	Synonyms	Description
PCT	% PERCENT PERCENTAGE	Calculates and lists percents.
TOTAL		Calculates and lists the field total. (The TOTAL keyword differs from the TOTAL function used with I-descriptor fields.)
TRANSPORT		Lists the last value for the field.

Field Modifier Keywords (Continued)

Field Qualifier Keywords

The following table lists keywords you can use to define inline field descriptors in Retrieve sentences. Field qualifiers temporarily override the file dictionary for the duration of the Retrieve command. For more information about field qualifiers, see the *Guide to Retrieve*.

Keyword	Description
AS	Specifies a name for an EVAL expression or a synonym for a field name.
ASSOC	Associates a field expression with an association of multivalued fields.
ASSOC.WITH	Associates a field expression with another multivalued field.
COL.HDG	Defines a column heading. Same as DISPLAY.NAME.
CONV	Defines a conversion.
DISPLAY.LIKE	Sets display characteristics.
DISPLAY.NAME	Defines a column heading. Same as COL.HDG.
FMT	Defines output format.
MULTI.VALUE	Specifies a multivalued field expression.
SINGLE.VALUE	Specifies a singlevalued field expression.

Field Qualifier Keywords

Report Qualifier Keywords

The following table lists keywords you can use to customize Retrieve reports.

Keyword	Synonym	Description
AUX.PORT		Sends output to a printer connected to terminal's auxiliary port.
COL.HDR.SUPP	COL-HDR-SUPP	Suppresses default report and column headings.
COL.SPCS	COL.SPACES	Changes default spacing between columns.
COL.SUP	COL-SUPP	Suppresses default column headings.
COUNT.SUP		Suppresses display of record count.
DBL.SPC	DBL-SPC	Double-spaces output records.
DET.SUP	DET-SUPP	Displays breakpoints only.
FIRST	SAMPLE	Limits number of records selected.
FOOTING	FOOTER	Defines footer for report page.
GRAND.TOTAL	GRAND-TOTAL	Specifies text for grand total line.
HDR.SUP	HDR-SUPP SUPP	Suppresses default report heading.
HEADING	HEADER	Overrides default heading.
ID.ONLY	ONLY	Displays record IDs only.
ID.SUP	ID-SUP ID-SUPP	Suppresses record ID display.
LPTR		Prints output on line printer.
MARGIN		Specifies size of left margin.
NO.SPLIT		Starts a record on a new page if it does not fit on current page.
NOPAGE	NO.PAGE	Suppresses automatic paging.

Report Qualifier Keywords

Keyword	Synonym	Description
ONLY	ID.ONLY	Displays record IDs only.
SAMPLE	FIRST	Limits the number of records selected.
SAMPLED		Limits the number of records selected.
SUPP	HDR.SUP HDR-SUPP	Suppresses default heading.
VERTICALLY	VERT	Displays output in vertical format.
Report Qualifier Keywords (Continued)		

Keyword Descriptions

The UniVerse keywords are listed and defined in this section.

(...)

Use in Retrieve sentences to group selection criteria. Parentheses alter the order of the evaluation.

@ASSOC_ROW

Use as a column name in the SQL SELECT, INSERT, UPDATE, and DELETE statements to reference an association's keys. @ASSOC_ROW generates a sequence of unique numbers which, when used with the primary key of the base table, produces a set of jointly unique association row keys. Use @ASSOC_ROW when the association does not have explicitly defined association keys. @ASSOC_ROW acts as a virtual column name when you dynamically normalize an NF² table or file.

For example, if a multivalued column for telephone numbers contains fax numbers as the fifth value, you can delete the fax numbers using a WHERE clause like the following to select only the fifth association row of each base table row:

```
>DELETE FROM CUSTOMERS_PHONES  
SQL+WHERE @ASSOC_ROW = 5;
```

1NF

Use with [SET.SQL](#) to turn first-normal-form mode on or off. This setting overrides the setting specified by the SQLSetConnectOption function.

32BIT

Use with [CREATE.FILE](#) and [RESIZE](#) to override the current setting of the 64BIT_FILES configurable parameter. 32BIT creates a 32-bit file on a UniVerse system using 64-bit file systems.

64BIT

Use with [CREATE.FILE](#) and [RESIZE](#) to override the current setting of the 64BIT_FILES configurable parameter. 64BIT creates a 64-bit file on a UniVerse system using 32-bit file systems.

A

Use in any Retrieve sentence to make it more English-like. This keyword is not required and is ignored by Retrieve.

–ACCEPT

Use with [MESSAGE](#) to enable receive mode for all messages.

ADDING

Use with [DEFINE.DF](#) to add a part file to a distributed file.

AFTER

Synonym for [GT](#).

ALL

Use with [ACCOUNT.FILE.STATS](#) to gather statistics for all files in an account.

Use with the secondary indexes commands [BUILD.INDEX](#), [DELETE.INDEX](#), [ENABLE.INDEX](#), [LIST.INDEX](#), and [UPDATE.INDEX](#), to specify that the command should affect all secondary indexes of the file.

Use with [CLEARSELECT](#) to clear all active select lists.

Use with [CONFIG](#) to list the information from the BRIEF and DATA keywords.

Use with **COPY** to copy all records in the source file. For example, to copy all records from FILE1 to FILE2, enter the following:

```
>COPY FROM FILE1 TO FILE2 ALL
```

Use with **MASTER** to release all task synchronous locks, enable the Break key for all users, or log out all users.

Use with **RECOVERY.CHECKPOINT** to check all recoverable files listed as currently activated in the UV.TRANS file.

Use with **UNLOCK** to remove all locks.

Use with **VERIFY.SQL** to verify all SQL objects on the system.

In NLS mode, use with **GET.LOCALE** to display all locale settings. Use with the **SET.LOCALE** command to set all categories for a locale. Use with **LIST.LOCALES** to list all defined locales. Use with **LIST.MAPS** to list all defined maps.

ALL.MATCH

Use with **SEARCH** to select only records containing all the specified strings.

AND

Logical operator AND used to join selection expressions, WHEN clauses, and IF statements. For example:

```
LIST MAIL WITH STATE EQ MA AND INTEREST EQ PAINT  
IF <<Enter INTEREST>> = PAINT AND STATE = MA THEN GO DO.IT:
```

ANY

Use in any Retrieve sentence to make it more English-like. This keyword is not required and is ignored by Retrieve.

APPEND

Use with **DIVERT.OUT** to append output from diversion to the end of an existing record.

ARCHIVE

Use with [SET.LOG.ATTR](#) to turn transaction logging archive mode on or off.

ARE

Use in any Retrieve sentence to make it more English-like. This keyword is not required and is ignored by Retrieve.

AS

Use with [SETPTR \(UNIX\)](#) or [SETPTR \(Windows Platforms\)](#) as a synonym for BANNER.

Use as a field qualifier in a Retrieve sentence or SQL SELECT statement to specify a new name for the field name or EVAL expression it qualifies. Its syntax is as follows:

```
AS temp.name
```

An AS clause follows the name of the field or the EVAL expression it qualifies. In the same Retrieve query you can use *temp.name* anywhere after the AS clause to refer to the field or EVAL expression by its new name. *temp.name* cannot be the name of an existing field in the data file or the name of a VOC entry.

–AS

Use with [SPOOL](#) to print an alias instead of a file name on the flag page.

ASSOC

Use as a field qualifier in a Retrieve sentence or SQL SELECT statement to associate a field expression with an existing association of multivalued fields. Its syntax is as follows:

```
ASSOC "association"
```

ASSOC temporarily overrides the file dictionary for the duration of the query. An ASSOC clause follows the name of the field or the EVAL expression it qualifies.

association is the record ID of the entry in the file dictionary that defines an association. It must be enclosed in single or double quotation marks. The entry defining the association must be in the file dictionary, not in the VOC file.

You cannot use the keywords ASSOC.WITH or SINGLE.VALUE in the same field expression with an ASSOC clause.

ASSOC.WITH

Use as a field qualifier in a Retrieve sentence or SQL SELECT statement to associate a field expression with another field expression that is multivalued. Its syntax is as follows:

$$\text{ASSOC.WITH } \{ \textit{fieldname} \mid \textit{temp.name} \}$$

ASSOC.WITH temporarily overrides the file dictionary for the duration of the query. An ASSOC.WITH clause follows the name of the field or EVAL expression it qualifies.

fieldname and *temp.name* must be multivalued.

fieldname is the name of a field defined in the file dictionary.

temp.name is the name of a field expression specified earlier in the same Retrieve query by an AS clause.

If the field expression qualified by the ASSOC.WITH clause is singlevalued, it is marked as multivalued for the duration of the query.

You cannot use the keywords ASSOC or SINGLE.VALUE in the same field expression with an ASSOC.WITH clause.

AT

Use with [CREATE.INDEX](#) to specify the account where you want to store secondary indexes you create.

Use with [SETPTR \(UNIX\)](#) or [SETPTR \(Windows Platforms\)](#) to route jobs to a system printer specified as its argument.

–AT

Use with **SPOOL** to specify the name of a printer.

AUTONL

Use with **SET.TERM.TYPE** to generate a newline sequence if a line longer than the width of the screen is typed.

AUX.PORT

Use in a Retrieve sentence or SQL SELECT statement to send results to a printer connected to the terminal's auxiliary port; results are also displayed on the screen.

The auxiliary port must be enabled by setting the *mc4* and *mc5* definitions in the *terminfo* entry for the user's terminal. The *mc4* definition turns off the auxiliary port, the *mc5* definition turns it on.

AUXMAP

In NLS mode, use with **SET.TERM.TYPE** to set a map for an auxiliary printer attached to the terminal.

AVERAGE

Synonym for AVG.

AVG

Use in a Retrieve sentence or SQL SELECT statement to calculate the average for a singlevalued or multivalued numeric field of the selected records and list it at the bottom of a report. Its syntax is as follows:

AVG field.expression [NO.NULLS]

field.expression specifies the name of a field or an EVAL expression for which you want to calculate the average value. Retrieve treats nonnumerics as zero values.

NO.NULLS tells Retrieve to ignore empty string values.

When you use the AVG keyword with a breakpoint keyword ([BREAK.ON](#) or [BREAK.SUP](#)), Retrieve lists breakpoint averages in addition to the overall average at the bottom of a report.

To average payments made in the last month, excluding empty string values, use a sentence like the following:

```
>LIST PAYABLES WITH PYMT.DATE >= 2/1/94 AVG PRICE NO.NULLS
```

BANNER

Use with [SETPTR \(UNIX\)](#) or [SETPTR \(Windows Platforms\)](#) to specify a name on the banner page. Its syntax is as follows:

```
BANNER [ NEXT | UNIQUE ] [ name ]
```

In mode 1, *name* appears on the second line of the banner page under the account name. In mode 3, specifies the record ID of the record in the [&HOLD&](#) file which stores the report. If you do not specify *name*, the record ID is P#0000. If you specify *name*, it is the record ID of the output record. In either case, each subsequent print job uses the same record ID and overwrites the previous job.

In mode 3, BANNER NEXT appends a sequential number to the name under which successive records are created in the [&HOLD&](#) file. If you do not specify *name*, the record ID is P#0000_#####, where ##### is increased for each new print job. If you specify *name*, the record ID is *name*#####.

In mode 3, BANNER UNIQUE appends a sequential number to the name under which successive records are created in the [&HOLD&](#) file. If you do not specify *name*, the record ID is P#0000_#####, where ##### is increased by each subsequent SETPTR command. If you specify *name*, the record ID is *name*#####.

BASE

Use with [LIST.MAPS](#) to recursively list any underlying base maps.

BASIC

Use with [HELP](#) to list help about BASIC statements and functions.

BCI

Use with [HELP](#) to list help about BASIC SQL Client Interface functions.

BEFORE

Synonym for [LT](#).

BLK

Use with [ASSIGN](#), [T.ATT](#), [T.DUMP](#), [REFORMAT](#), and [SREFORMAT](#) to define the block size of a tape record. The default block size is 8192 bytes, the maximum size. When you restore tape records to disk using [T.LOAD](#), the block size is displayed.

For example:

```
>T.DUMP ACCOUNTS BLK 8000
```

BLOCK

Use with [CONNECT](#) to define how to terminate input statements.

BREAK

Use with [MASTER](#) to enable the Break key.

BREAK-ON

Synonym for [BREAK.ON](#).

BREAK.ON

Use with Retrieve commands and the SQL SELECT statement to specify which field to use to create breaks in a report. A break occurs when the column values change. The break is indicated by asterisks or by user-specified text. BREAK.ON expressions are often used with the keywords AVG, CALC, PCT, and TOTAL to perform the specified action and display the results when the values change.

To make the report more effective, sort the breakpointed field to process and display the same values together. If the field is multivalued, specify an exploded sort (using the BY.EXP or BY.EXP.DSND keyword).

The syntax for the breakpoint expression is as follows:

```
BREAK.ON ["[text] ['options'] ..." ] field [qualifiers]
```

text is any text you want to appear under the field value in the breakpoint line. If you specify *text*, the row of asterisks is suppressed. Text is not displayed when the report is in vertical format.

options can be any of the following formatting options. Options must be contained within single quotation marks, such as 'BDL'. All options suppress the breakpoint row of stars.

Option	Description
B	Use with the B option of the HEADING or FOOTING keyword to include the current breakpoint value in the heading or footing. Every time the breakpoint value changes, a new page is generated. Only the first B option in a sentence is used.
D	Suppresses printing of the breakpoint line if there is only one line of detail for a specific value, but leaves a blank line between records.
L	Suppresses printing of the breakpoint line, but still skips a line when the value changes. If any text is specified in the sentence, it is ignored.
N	Resets the page number to 1 for each new breakpoint value.
O	Outputs each breakpoint value only once.
P	Begins a new page for every new breakpoint value.
V	Inserts the breakpoint field value instead of the line of stars.

BREAK.ON Options

Enclose the entire first argument, including all text and options, in double quotation marks.

field is the field to break on.

The breakpoint line in a report is a row of stars in the column displaying the breakpoint field and a row of dashes in any column being totalled, averaged, calculated, or having its percentage calculated.

For example, if you have a PRODUCT field in a PAYABLES file and want subtotals of money spent by product, you might create a sentence such as:

```
>LIST PAYABLES BY PRODUCT BREAK.ON PRODUCT TOTAL UNIT.PRICE
```

BREAK.SUP

Use with Retrieve commands and the SQL SELECT statement to specify which field to use to create breaks in a report. BREAK.SUP does the same thing as [BREAK.ON](#), except it does not display the row of stars or a column containing the values in the field specified.

The syntax for the breakpoint expression is as follows:

```
BREAK.SUP [ " [ text ] [ 'options' ] ..." ] field [ qualifiers ]
```

BREAK.SUP uses the following options:

Option	Description
B	Use with the B option of the HEADING and FOOTING keywords to include the current breakpoint value in the heading or footing. Every time the breakpoint value changes, a new page is generated. Only the first B option in a sentence is used.
D	Suppresses the entire breakpoint line if there is only one line of detail for a specific value.
P	Begins a new page for every new breakpoint value.

BREAK.SUP Options

For example:

```
>LIST PAYABLES BY PRODUCT BREAK.SUP PRODUCT TOTAL UNIT.PRICE
```

BRIEF

Use with [CONFIG](#) to list the license number, the licensed number of users for the system, and the license expiration date.

Use with [CONVERT.SQL](#) to suppress the list of column and association definitions.

Use with [SETPTR \(UNIX\)](#) or [SETPTR \(Windows Platforms\)](#) to suppress display of SETPTR settings for approval.

Use with the [PHANTOM](#) command to direct process output to the null device.

BY

Use in a sort expression to sort selected records. Records are sorted in ascending order of values in the specified field. Its syntax is as follows:

`BY field`

field is a singlevalued field. If you use BY with a multivalued field, value marks are ignored, and the values are treated as a single field and sorted as a unit. To sort by values in a multivalued field, use the [BY.EXP](#) keyword.

Fields with an R in the FORMAT field of the data descriptor are sorted right-justified. Fields with an L or a T in the FORMAT field are sorted left-justified.

You can use more than one sort expression in a Retrieve query. Retrieve processes sort expressions from left to right, so the first sort field is the primary sort key, the second sort field is the secondary sort key, and so on.

To create a list of balances from the smallest to the largest balance, use a sentence like the following:

```
>SELECT PAYABLES BY BAL.DUE
```

To display last names in alphabetical order, use a sentence like the following:

```
>SORT MAIL BY LNAME
```

BY-DSND

Synonym for BY.DSND.

BY.DSND

Use in a sort expression to sort selected records. Records are sorted in descending order of values in the specified field. Its syntax is as follows:

`BY.DSND field`

field is a singlevalued field. If you use BY.DSND with a multivalued field, value marks are ignored, and the values are treated as a single field and sorted as a unit. To sort by values in a multivalued field, use the [BY.EXP.DSND](#) keyword.

Fields with an R in the FORMAT field of the data descriptor are sorted right-justified. Fields with an L or a T in the FORMAT field are sorted left-justified.

You can use more than one sort expression in a Retrieve query. Retrieve processes sort expressions from left to right, so the first sort field is the primary sort key, the second sort field is the secondary sort key, and so on.

To display account numbers from the highest number to the lowest number, use a sentence like the following:

```
>SORT ACCOUNTS BY.DSND ACCT.NO
```

BY-EXP

Synonym for BY.EXP.

BY.EXP

Use in a sort expression to explode and sort data in multivalued fields. The exploded multivalues are sorted in ascending order. Its syntax is as follows:

```
BY.EXP mvfield
```

mvfield is a multivalued field. If you use BY.EXP on a singlevalued field, the field is sorted as if BY were specified.

Fields with an R in the FORMAT field of the data descriptor are sorted right-justified. Fields with an L or a T in the FORMAT field are sorted left-justified.

Multiple Exploded Sort Expressions

You can use more than one exploded sort expression in a Retrieve query. Retrieve processes sort expressions from left to right, so the first sort field is the primary sort key, the second sort field is the secondary sort key, and so on.

To use more than one exploded sort expression (beginning with BY.EXP or BY.EXP.DSND) in a Retrieve query, the fields specified by *mvfield* must belong to the same association. If they belong to different associations, Retrieve uses only the first BY.EXP or BY.EXP.DSND clause and ignores all other exploded sort expressions. If the exploded fields do not belong to any association, Retrieve assumes a default association for the duration of the query.

If the exploded field contains only subvalues, the subvalues are treated as if they were values. If the exploded field contains values and subvalues, and if the structures of the associated fields are the same, Retrieve explodes the data to the subvalue level and sorts as expected. If the structures of the associated fields are different, the following rules apply:

- If the field specified by the first exploding sort expression contains values only, Retrieve explodes only to the value level, regardless of whether fields specified by subsequent sort expressions contain subvalues. If the first exploding sort field contains subvalues, Retrieve explodes to the subvalue level.
- Each atomic value of the first field is matched against corresponding values in fields specified by subsequent sort expressions, regardless of whether the corresponding values contain subvalues. Each subvalue of the first field is exploded and matched against corresponding subvalues in fields specified by subsequent sort expressions.
- If the associated fields do not have a corresponding value or subvalue, Retrieve supplies an empty string. If the first field is missing a value or subvalue corresponding to a value or subvalue in the associated fields, Retrieve supplies an empty string.

Exploded Sorts and Select Lists

If you use BY.EXP in a **SELECT** or **SSELECT** query, the format of the select list differs from the standard format. Each element in the select list has the following format:

*record.id***v***value***s***field***v***subvalue*

record.id is the record ID where the data value is found. *field*, *value*, and *subvalue* are the field, value, and subvalue numbers within the record containing the data value. **v** represents a value mark, and **s** represents a subvalue mark. If you use more than one BY.EXP or BY.EXP.DSND clause in a SELECT or SSELECT query, *field* is the number of the field specified in the first BY.EXP clause.

To display a list of products, use a sentence like the following:

```
>SORT PAYABLES BY.EXP PRODUCTS
```

BY-EXP-DSND

Synonym for BY.EXP.DSND.

BY.EXP.DSND

Use in a sort expression to explode and sort data in multivalued fields. The exploded multivalues are sorted in ascending order. Its syntax is as follows:

```
BY.EXP.DSND mvfield
```

mvfield is the multivalued field. If you use BY.EXP.DSND on a singlevalued field, the field is sorted as if BY.DSND were specified.

Fields with an R in the FORMAT field of the data descriptor are sorted right-justified. Fields with an L or a T in the FORMAT field are sorted left-justified.

Multiple Exploded Sort Expressions

You can use more than one exploded sort expression in a Retrieve query. Retrieve processes sort expressions from left to right, so the first sort field is the primary sort key, the second sort field is the secondary sort key, and so on.

To use more than one exploded sort expression (beginning with BY.EXP or BY.EXP.DSND) in a Retrieve query, the fields specified by *mvfield* must belong to the same association. If they belong to different associations, Retrieve uses only the first BY.EXP or BY.EXP.DSND clause and ignores all other exploded sort expressions. If the exploded fields do not belong to any association, Retrieve assumes a default association for the duration of the query.

If the exploded field contains only subvalues, the subvalues are treated as if they were values. If the exploded field contains values and subvalues, and if the structures of the associated fields are the same, Retrieve explodes the data to the subvalue level and sorts as expected. If the structures of the associated fields are different, the following rules apply:

- If the field specified by the first exploding sort expression contains values only, Retrieve explodes only to the value level, regardless of whether fields specified by subsequent sort expressions contain subvalues. If the first exploding sort field contains subvalues, Retrieve explodes to the subvalue level.

- Each atomic value of the first field is matched against corresponding values in fields specified by subsequent sort expressions, regardless of whether the corresponding values contain subvalues. Each subvalue of the first field is exploded and matched against corresponding subvalues in fields specified by subsequent sort expressions.
- If the associated fields do not have a corresponding value or subvalue, Retrieve supplies an empty string. If the first field is missing a value or subvalue corresponding to a value or subvalue in the associated fields, Retrieve supplies an empty string.

Exploded Sorts and Select Lists

If you use BY.EXP.DSND in a SELECT or SSELECT query, the format of the select list differs from the standard format. Each element in the select list has the following format:

```
record.idvvaluesfieldvsubvalue
```

record.id is the record ID where the data value is found. *field*, *value*, and *subvalue* are the field, value, and subvalue numbers within the record containing the data value. **v** represents a value mark, and **s** represents a subvalue mark. If you use more than one BY.EXP or BY.EXP.DSND clause in a SELECT or SSELECT query, *field* is the number of the field specified in the first BY.EXP clause.

CALC

Use in a Retrieve sentence or SQL SELECT statement to specify total calculations in I-descriptors. CALC can be used with breakpointing to produce subtotals. Its syntax is as follows:

```
CALC field
```

field is the name of an I-descriptor that is one of the fields included in the report. The I-type expression must include the TOTAL function.

When CALC is used in a sentence with breakpointing, intermediate values for the expression are displayed on the breakpoint lines. The individually accumulated subtotals are used to calculate an intermediate value for the entire expression at the breakpoint. A final value for the expression is printed at the bottom of the report.

The ACCOUNTS file contains an I-descriptor called DIFFERENCE. It looks like this:

```
TOTAL (AMT . RCD) -TOTAL (AMT . PD)
```

A sentence using the I-descriptor and CALC looks like this:

```
>LIST ACCOUNTS BY ACCT.NO BREAK.ON ACCT.NO AMT.PD AMT.RCD CALC  
DIFFERENCE
```

CALCULATE

Synonym for CALC.

CANCEL

Use with [DEFINE.DF](#) to remove the part file number and algorithm from a part file.

-CANCEL

Use with [SPOOL](#) to delete a print job from the printer queue.

CATALOG

Use with [VERIFY.SQL](#) to verify the internal consistency of the SQL catalog.

CHECKPOINT

Use with [SET.LOG.ATTR](#) to turn transaction logging checkpoint mode on or off

CLEAR

Use with [ENVIRONMENT](#) or [ENV](#) to clear an environment variable setting.

COL.HDG

Use as a field qualifier in a Retrieve sentence or SQL SELECT statement to define a column heading for the field or EVAL expression it qualifies. Its syntax is as follows:

```
COL.HDG "heading"
```

COL.HDG temporarily overrides the file dictionary for the duration of the query. A COL.HDG clause follows the name of the field or EVAL expression it qualifies.

heading is the text of the heading at the top of the column when the field is displayed. The heading must be enclosed in single or double quotation marks.

To specify a line break in the column heading, use the letter L enclosed in single quotation marks and enclose *heading* in double quotation marks. If you specify *heading* as an empty string, the field name (for example, the record ID of the field definition in the dictionary) is used.

COL-HDR-SUPP

Synonym for COL.HDR.SUPP.

COL.HDR.SUPP

Use in a Retrieve sentence or SQL SELECT statement to suppress printing of the default page heading (page, time, and date), and column headings. Used in LIST.LABEL and SORT.LABEL statements, COL.HDR.SUPP produces a continuous form report without page breaks. The COL.HDR.SUPP phrase is a combination of the COL.SUP and HDR.SUP keywords.

COL.SPACES

Synonym for COL.SPCS.

COL.SPCS

Use in a Retrieve sentence or SQL SELECT statement to change the default spacing between columns in the display. The default varies from one through four depending on the total width of the fields being displayed and the length of the field names. Each space is the width of a single screen column.

To change the spacing, include the number of spaces after the keyword COL.SPCS. If you do not include a number, only one space is left between columns. For example:

```
>LIST PAYABLES COL.SPCS 5
```

COL-SUPP

Synonym for COL.SUP.

COL.SUP

Use in a Retrieve sentence or SQL SELECT statement to suppress the column headings, which are the display names that normally display at the top of the column. For example:

```
>LIST PAYABLES VENDORS BAL.DUE WITH BAL.DUE GT 10 COL.SUP
```

COLLATE

In NLS mode, use with the [CREATE.INDEX](#) command to specify the name of the locale whose Collate convention you want to associate with an index. Use with [GET.LOCALE](#) and [SET.LOCALE](#) to specify the category value.

COMPLETE

Use with [CATALOG](#) to specify that the VOC entry for a locally cataloged program is to contain the program's absolute pathname. The VOC entry normally contains the location of the program relative to the user's account.

CONCURRENT

Use with [RESIZE](#) to permit other users to access the file while it is being resized. To let other users use the file PAYABLES while it is being resized, enter the following:

```
>RESIZE PAYABLES 4 3 2 CONCURRENT
```

CONV

Use as a field qualifier in a Retrieve sentence or SQL statement to define a conversion for the field name or EVAL expression it qualifies. Its syntax is as follows:

```
CONV "conversion"
```

CONV temporarily overrides the file dictionary for the duration of the query. A CONV clause always follows the name of the field or EVAL expression it qualifies.

conversion is any BASIC conversion code available to the ICONV and OCONV functions. The conversion must be enclosed in single or double quotation marks. If there is a conversion in the dictionary entry and you want no conversion applied, specify an empty string in the CONV clause.

Use with [HELP](#) to list help for BASIC correlatives and conversions.

COPIES

Use with [SETPTR \(UNIX\)](#) or [SETPTR \(Windows Platforms\)](#) to print the number of copies specified in its argument of each report with a single banner page.

–COPIES

Use with [SPOOL](#) to specify the number of copies to print.

COUNT.SUP

Use in a Retrieve sentence or SQL SELECT statement to suppress the message that lists the number of records processed.

Use with [MERGE.LIST](#) to suppress the display of the number of records selected.

CREATE

Use with [CONVERT.SQL](#) to analyze a file dictionary or the SQLDEF file, list column and association definitions, and convert a file to a table.

CRT

Use with [LIST.DIFF](#), [LIST.INTER](#), or [LIST.UNION](#) to copy data to the terminal.

CTYPE

Use with [SET.LOCALE](#) to set the CTYPE category.

DATA

Use with [CONFIG](#) to list the current values of the UniVerse configurable parameters.

Use with a file name to specify the data file. Specify DATA immediately before the file name. To delete only the data file OVERDUE but retain its dictionary, enter the following:

```
>DELETE.FILE DATA OVERDUE
```

DBL-SPC

Synonym for DBL.SPC.

DBL.SPC

Use in a Retrieve sentence or SQL SELECT statement to set double spacing between records in a report, overriding the default single spacing. For example:

```
>LIST MAIL WITH ZIP LIKE 02... DBL.SPC
```

DEFAULT

Use with [SET.TERM.TYPE](#) to set the map for the corresponding terminal type from its entry in the *terminfo* directory.

Use with the [ASSIGN](#), [SET.FILE.MAP](#), [SET.GCI.MAP](#), [SET.SEQ.MAP](#), and [T.ATT](#) commands to specify the map name designated by the corresponding configurable parameter in the *uvconfig* file.

DEFAULTS

Use with [CONFIGURE.FILE](#) to configure a dynamic (type 30) file with the default dynamic file parameters. The values for GROUP.SIZE and RECORD.SIZE are not changed by the DEFAULTS keyword.

DEFER

Use with [SETPTR \(UNIX\)](#) or [SETPTR \(Windows Platforms\)](#) to defer printing of the report until the time specified in its argument. Its argument can be specified in one of the following formats:

```
hh:mm  
dd.hh:mm  
mm.dd.hh:mm  
yy.mm.dd.hh:mm  
dd  
mm.dd  
yy.mm.dd  
+mm  
+hh:mm  
+dd.hh:mm
```

DELETE

Use with [COMO](#) to delete a COMO record.

DELETING

Use with [COPY](#) to delete the records from the source file after they are copied to the new file. For example, to delete records 101 and 102 from the ACCOUNTS file after copying them to the DEADACCOUNTS file, enter the following:

```
>COPY FROM ACCOUNTS TO DEADACCOUNTS 101 102 DELETING
```

DET-SUPP

Synonym for DET.SUP.

DET.SUP

Use in a Retrieve sentence or SQL SELECT statement containing a [BREAK.ON](#) expression to suppress detail lines and display only breakpoint, subtotal, and total lines. For example:

```
>LIST PAYABLES BY PRODUCT BREAK.ON PRODUCT TOTAL UNIT.PRICE  
DET.SUP
```

DETAIL

Use with [LIST.INDEX](#) to display additional information detailing the name of each secondary key value, the number of records with that key value, and the bytes used for that key value.

Use with [LIST.LOCALES](#) to provide detailed information about each locale.

Use with [LIST.MAPS](#) to list the map definitions in the NLS.MAPS.DESCS file.

DEVICE

Use with [PORT.STATUS](#) to limit the report to jobs attached to a specific device.

Use with [PTERM \(Windows Platforms\)](#) to specify device characteristics.

Use with [UNLOCK](#) to restrict lock removal to a specific file or device.

DEVICELIST

Use with [SET.LOG.ATTR](#) to specify a list of tape devices to which to log transaction updates.

DICT

Use with a file name to specify the file dictionary rather than the data file. Specify DICT immediately before the file name. For example:

```
>REVISE DICT PAYABLES  
>LIST DICT PAYABLES WITH TYPE EQ PH
```

DIFF

Use with [MERGE.LIST](#) to produce a select list whose elements are the remainder of *list1* after the elements of *list2* that are also in *list1* are subtracted from it.

DISABLE LOCK.HIST

Use with [PORT.STATUS](#) to disable logging of concurrency control operations.

DISKS

Use with [STATUS](#) to display disk usage information about all disks.

DISPLAY

Use with [ENVIRONMENT](#) or [ENV](#) to list the current environment settings.

Use with [PTERM \(UNIX\)](#) or [PTERM \(Windows Platforms\)](#) to display a list of the current terminal characteristics.

DISPLAY.LIKE

Use as a field qualifier in a Retrieve sentence or SQL SELECT statement to set a field's display characteristics to be the same as those of another field. Its syntax is as follows:

```
DISPLAY.LIKE { fieldname | temp.name }
```

A DISPLAY.LIKE clause follows the name of an existing field or an EVAL expression.

fieldname is the name of a field defined in the file dictionary.

temp.name is the name of a field expression specified earlier in the same Retrieve query by an AS clause.

When used in the same field expression with other field qualifiers, a DISPLAY.LIKE clause is processed before the CONV, COL.HDG, DISPLAY.NAME, FMT, SINGLE.VALUE, MULTI.VALUE, ASSOC, and ASSOC.WITH field qualifiers. You can use any of these keywords to override display characteristics set by the DISPLAY.LIKE clause.

DISPLAY.NAME

Synonym for [COL.HDG](#).

DOWN

Use with [CHAP](#) to lower the priority of processing tasks for that account. CHAP DOWN causes 10 to be added to the priority number assigned each task.

DYNAMIC

Use with [CREATE.FILE](#), [CONFIGURE.FILE](#), and [RESIZE](#) to specify a dynamic file. Dynamic files are created as type 30 files. DYNAMIC and 30 are equivalent file types.

EJECT

Use with [SETPTR \(UNIX\)](#) or [SETPTR \(Windows Platforms\)](#) to perform a page eject at the end of a print job.

EMPTY.NULL

Use with [SET.SQL](#) to turn empty-null mapping on or off.

ENABLE LOCK.HIST

Use with [PORT.STATUS](#) to enable logging of concurrency control operations.

ENDPAGE

Use with [SETPTR \(UNIX\)](#) or [SETPTR \(Windows Platforms\)](#) to specify ending page number for printing.

ENUM

Use in a Retrieve sentence to count the number of values that occur in a specified field in a set of records. Its syntax is as follows:

```
ENUM field.expression [NO.NULLS]
```

field.expression specifies the name of a field or an EVAL expression for which you want to calculate the count. Retrieve treats nonnumerics as zero values.

NO.NULLS tells Retrieve to ignore empty string values.

If the field uses the default column header, the ENUM keyword is added before the column header.

If you use the ENUM keyword with a breakpoint keyword ([BREAK.ON](#) or [BREAK.SUP](#)), Retrieve lists the data value for each record in addition to the enumeration count for each breakpoint and the final enumeration count at the bottom of the list.

If you specify a multivalued field, Retrieve counts each value separately.

For example, the following sentence displays each phone number in the records of the PERSONNEL file, excluding empty string values, listing the number of values at the bottom of the report:

```
>LIST PERSONNEL ENUM PHONE NO.NULLS
```

ENUMERATE

Synonym for ENUM.

EQ

The relational operator EQUAL, used in selection expressions, WHEN clauses, and IF statements. For example:

```
>SORT INVENTORY WITH ITEM EQ 'SHIRTS'
```

EQUAL

Synonym for EQ.

EVAL

Use in any Retrieve sentence to introduce an in-line I-type expression. An EVAL expression defines a new virtual field which exists only for the duration of the current query. Its syntax is as follows:

```
EVAL "i.type.expr" [ qualifiers ]
```

i.type.expr is any expression that can be used in an I-descriptor, enclosed in single or double quotation marks.

qualifiers can be one or more field qualifier clauses beginning with the following keywords: [AS](#), [ASSOC](#), [ASSOC.WITH](#), [COL.HDG](#), [CONV](#), [DISPLAY.LIKE](#), [DISPLAY.NAME](#), [FMT](#), [MULTI.VALUE](#), and [SINGLE.VALUE](#).

You can use an EVAL expression anywhere you use a field name: in selection and sort expressions, as part of an output specification, and so on.

When the field is output, the following default conditions apply:

- The column heading is the text of the I-type expression, unless you use the [COL.HDG](#) or [DISPLAY.NAME](#) keywords to specify a column heading. If the text is longer than 20 characters, value marks are inserted in the text every 20 characters or at the width defined by the display format, whichever is greater. This creates a multiline column heading.
- The conversion, display format, single- or multivalued code, and association specifications are derived from the dictionary entry that defines the first field appearing in the I-type expression.
- If no field appears in the I-type expression, the temporary field has no conversion, its format is 10L, and it is singlevalued.

EVALUATE

Synonym for EVAL.

EVERY

Use in a selection expression to select a record only if every value in a multivalued field meets the specified condition. EVERY must be used with the [WITH](#) keyword. For example:

```
>LIST INVENTORY WITH EVERY ITEM EQ 'PAPER'
```

Use with [LIST.READU](#) to list active group locks in addition to the active file and record locks.

EXPLODE

Use with [SEARCH](#) to specify that an element be included in the select list for each value in a record that matches a specified string. The select list elements also contain the field and value number in which the string was found. When the EXPLODE keyword is used, the list elements are created according to the form:

```
record.idVvalueSfieldV0
```

record.id is the record ID where the string is found.

field and *value* are the field and value numbers within the record containing the string.

v represents a value mark, and **s** represents a subvalue mark.

EXTERNAL

Use with **DEFINE.DF** and **REBUILD.DF** to specify an external routine as the partitioning algorithm for a distributed file.

FILE

Use in any Retrieve sentence to make it more English-like. This keyword is not required and is ignored by Retrieve.

Use with **UNLOCK** to restrict lock removal to a specific file or device.

FILE.OFF

Use with **DIVERT.OUT** to suspend the diversion of command output to the current file. Subsequent command output is not diverted until a **DIVERT.OUT FILE.ON** command is entered.

FILE.ON

Use with **DIVERT.OUT** to resume sending command output to a file after it had been suspended by issuing a **DIVERT.OUT FILE.OFF** command.

FILELOCK

Use with **UNLOCK** to restrict lock removal to file locks. When combined with the **SEMAPHORE *n*** argument of the **UNLOCK** command, **FILELOCK** releases concurrency control semaphores that control access to the file lock table.

FILEMAP

Use with **PORT.STATUS** to generate a list of all files currently being used by the specified job.

FIRST

Use in a Retrieve sentence or SQL SELECT statement as a synonym for SAMPLE.

Use with the **COPY** command to copy the first *n* records from the source file. Its syntax is as follows:

```
FIRST n
```

FIX

Use with **VERIFY.SQL** to fix all inconsistencies in the SQL catalog.

FMT

Use with **SETPTR (UNIX) or SETPTR (Windows Platforms)** to specify that the spooler controls pagination and formatting.

Use as a field qualifier in a Retrieve sentence or SQL statement to define a format for the field name or EVAL expression it qualifies. Its syntax is as follows:

```
FMT "format"
```

FMT temporarily overrides the file dictionary in a Retrieve sentence or SQL SELECT statement. An FMT clause follows the name of the field or EVAL expression it qualifies.

format specifies the width of the display column, the character used to pad the display field, the type of justification, the format of numeric data, and a format mask. *format* must be enclosed in single or double quotation marks. For full details about the syntax of the format expression, see *UniVerse BASIC*.

If you specify *format* as an empty string, a default format of 10L is used. Invalid format expressions can give unpredictable results.

If you use FMT with a sort expression, Retrieve sorts the data according to the field format specified in the FMT expression.

FOOTER

Synonym for FOOTING.

FOOTING

Use in a Retrieve sentence or SQL SELECT statement to define a footer for the bottom of each page of your report. Its syntax is as follows:

`FOOTING " [ tDruext ] [ 'options' ] [ text ] ... "`

text is text you want to appear in the footing. You can intersperse options enclosed in single quotation marks anywhere in the text.

options can be any of the following:

Option	Description
B[n]	Inserts the current breakpoint field value in a field of <i>n</i> spaces when used with the B option of BREAK.ON . Each new value generates a new page.
C[n]	Centers the footing in a field of <i>n</i> spaces.
D	Inserts the current date. In NLS mode, the date format is controlled by the Time convention.
F[n]	Inserts the file name left-justified in a field of <i>n</i> spaces.
G	Inserts gaps in the footing format.
I[n]	Inserts the record ID left-justified in a field of <i>n</i> spaces. Same as R.
L	Inserts a carriage return and linefeed to make a multiple-line footing.
N	Suppresses page pause during a terminal display.
P[n]	Inserts the page number left-justified in a field of <i>n</i> spaces. The keyword begins with page 1 and adds 1 for each successive page.
Q	Lets you use the characters] (right bracket), ^ (caret), and \ (backslash) in footing text.
R[n]	Inserts the record ID left-justified in a field of <i>n</i> spaces. Same as I.
S	Inserts the page number left-justified. One space is reserved for the number. If the number of digits exceeds 1, text to the right of the number is shifted right by the number of extra digits.
T	Inserts the current time and date. In NLS mode, the time format is controlled by the Time convention

FOOTING Options

Text and options can be in any order. FOOTING displays the information in the order specified. You can use spaces anywhere in the footing text to improve readability. A space is equivalent to the width of a single screen column. The text and options must be enclosed in double quotation marks. To create a report with a footer, use a sentence like the following:

```
>LIST PAYABLES WITH PYMT.DATE <= 12/31/84 FOOTING "DECEMBER  
PAYMENTS 'D' 'P'"
```

Use the G option to add spaces to text in footings to bring the width of the line up to the device width. If you specify G once, spaces are added at that point in the line. If you specify G more than once, spaces are distributed as evenly as possible at every point where you put a G. For example:

Specification	Result
"Hello there"	Hello there
"'G'Hello there"	 there Hello
"'G'Hello there'G'"	Hello there
"Hello'G'there"	Hello there
"'G'Hello'G'there'G' "	Hello there

G Option

If the text includes an apostrophe ('), right bracket (]), caret (^), or backslash (\) that you want to include in the report, type an apostrophe before the character. For example:

```
>LIST PAYABLES WITH DUE DATE <= 1/1/85 FOOTING "JANUARY'S  
PAYMENTS 'LDP'"
```

You can also specify the Q option before the first occurrence of any of these characters. You only need to specify Q once in any heading. For example:

```
FOOTING "'Q'Using ], ^, and \ in Footings"
```

FOR

Use in any Retrieve sentence to make it more English-like. This keyword is not required and is ignored by Retrieve.

FORCE

Use with [CATALOG](#) to force the replacement of an existing global catalog routine of the same name. To replace the cataloged program RECEIVABLES with an updated version, enter the following:

```
>CATALOG BP *RECEIVABLES FORCE
```

Use in any Retrieve sentence to force the display of column headings and headers when no records are selected.

Use with the [DEFINE.DF](#) command to force a part file's number and algorithm to change even if the part file belongs to more than one distributed file.

Use with [SET.INDEX](#) to change settings without prompting.

In NLS mode, use with the [SET.FILE.MAP](#) command to set a new map for a file.

FORM

Use with [SETPTR \(UNIX\)](#) or [SETPTR \(Windows Platforms\)](#) to specify a special form for printing this report. The system operator is prompted to put the named form in the printer. The name can be up to six characters long, and is specified as an argument to FORM.

–FORM

Use with [SPOOL](#) to specify a form to use for printing.

FORM.FEED

Use with [LIST.ITEM](#) and [SORT.ITEM](#) to make each record begin on a new page.

–FORM.FEED

Use with [CP](#) and [CT](#) to list each record on a separate page.

FORMAT.MAP

In NLS mode, use with [SETPTR \(UNIX\)](#) or [SETPTR \(Windows Platforms\)](#) to set a map name for formatting only. Data still goes to the spool queue with a map name of NONE.

FROM

Use in a Retrieve sentence or in an [NSELECT](#), [QSELECT](#), or [SAVE.LIST](#) sentence to specify that the records listed in the specified numbered select list should be used. Its syntax is as follows:

```
FROM n
```

n is a number from 0 through 10 specifying which select list to use. To list the records from INVENTORY whose record IDs are stored in active select list 3, enter the following:

```
>LIST INVENTORY FROM 3
```

Use with [COPY](#) to specify the file from which to copy records.

Use with [HELP](#) to specify one of the system help files.

Use with [T.READ](#) to specify the tape record at which to start reading.

FTN

Use with [SETPTR \(UNIX\)](#) or [SETPTR \(Windows Platforms\)](#) to specify that reports contain FORTRAN control codes.

FUNDAMENTAL

Use with [SET.TERM.TYPE](#) to load the default key bindings.

GE

The relational operator GREATER THAN OR EQUAL TO used in selection expressions, WHEN clauses, and IF statements. For example:

```
>SORT INVENTORY WITH NUMBER GE 100
```

GEN

Use with [CONVERT.SQL](#) to generate a new SQLDEF file even if an old one exists.

GENERAL

Use with [CREATE.FILE](#), [CONFIGURE.FILE](#), and [RESIZE](#) to specify the general hashing algorithm for dynamic (type 30) files. The [SEQ.NUM](#) keyword can be used in place of GENERAL to specify a hashing algorithm suitable for sequential numbers. GENERAL is the default hashing algorithm for dynamic files.

GRAND-TOTAL

Synonym for GRAND.TOTAL.

GRAND.TOTAL

Use with Retrieve commands and the SQL SELECT statement to specify text to be printed on the grand total line of a report. Any text you specify is printed left-justified in the first column of the report. Text that exceeds the width specification for the first column is truncated. Its syntax is as follows:

```
GRAND.TOTAL "[text]" ['options']..."
```

options are the following:

Option	Description
L	Suppresses the double bar line above the grand total line.
P	Prints the double bar line and grand total line on a separate page.

GRAND.TOTAL Options

The following sentence specifies a grand total expression:

```
>LIST SUN.SPORT BY MEMBER.ID BREAK.ON MEMBER.ID TOTAL COST  
GRAND.TOTAL "Final Total"
```

GREATER

Synonym for GT.

GROUP

Use with [SET.REMOTE.ID](#) to limit a user's group membership on a remote UNIX system to a specific group.

Use with [UNLOCK](#) to restrict lock removal to the group specified by its group address. Its syntax is as follows:

```
GROUP group#
```

GROUP.SIZE

Use with [CREATE.FILE](#), [CONFIGURE.FILE](#), and [RESIZE](#) to specify the size of each group in a dynamic (type 30) file. Its syntax is as follows:

```
GROUP.SIZE { 1 | 2 }
```

1, the default, specifies a group size of 2048 bytes, which is equivalent to a separation of 4. The 2 specifies a group size of 4096 bytes, which is equivalent to a separation of 8.

GROUPLock

Use with [UNLOCK](#) to restrict lock removal to group locks and update record locks associated with the group. When combined with the SEMAPHORE *n* argument of the UNLOCK command, releases concurrency control semaphores that control access to the group lock table.

GT

The relational operator GREATER THAN used in selection expressions, WHEN clauses, and IF statements. For example:

```
>SORT PAYABLES WITH PYMT GT 50
```

HDR-SUPP

Synonym for HDR.SUP.

HDR.SUP

Use in a Retrieve sentence or SQL SELECT statement to suppress the default header, which contains the sentence that you executed, the time, the date, and the page number. For example:

```
>SORT MAIL WITH STATE EQ MA HDR.SUP LPTR
```

–HEAD

Use with [SPOOL](#) to print the flag page.

HEADER

Synonym for HEADING.

HEADING

Use in a Retrieve sentence or SQL SELECT statement to define a header at the top of each page of your report. This heading overrides the default heading. The default heading includes the sentence, the time, the date, and a page number. Its syntax is as follows:

```
HEADING "[text] ['options'] [text]..."
```

text is text to appear in the heading. You can intersperse options enclosed in single quotation marks anywhere in the text.

options can be any of the following:

Option	Description
B[n]	Inserts the current breakpoint field value in a field of <i>n</i> spaces when used with the B option of BREAK.ON . Each new value generates a new page.
C[n]	Centers the heading in a field of <i>n</i> spaces.
D	Inserts the current date. In NLS mode, the date format is controlled by the Time convention.
F[n]	Inserts the file name left-justified in a field of <i>n</i> spaces.
G	Inserts gaps in the heading format.
I[n]	Inserts the record ID left-justified in a field of <i>n</i> spaces. Same as R.
L	Inserts a carriage return and linefeed to make a multiple-line heading.
N	Suppresses page pause during a terminal display.
P[n]	Inserts the page number left-justified in a field of <i>n</i> spaces. The keyword begins with page 1 and adds 1 for each successive page.
Q	Lets you use the characters] (right bracket), ^ (caret), and \ (backslash) in heading text.
R[n]	Inserts the record ID left-justified in a field of <i>n</i> spaces. Same as I.
S	Inserts the page number left-justified. One character space is reserved for the number. If the number of digits exceeds 1, text to the right of the number is shifted right by the number of extra digits.
T	Inserts the current time and date. In NLS mode, the time format is controlled by the Time convention

HEADING Options

Text and options can appear in any order. HEADING displays the information in the order specified. You can use spaces anywhere in the heading text to improve readability. A space is equivalent to the width of a single screen column. The text and options must be enclosed in double quotation marks. To create a report with a header, use a sentence like the following:

```
>LIST PAYABLES WITH DUE.DATE <= 1/1/85 HEADING "JANUARY PAYMENTS  
'L' 'D' 'P'"
```

Use the G option to add spaces to text in headings to bring the width of the line up to the device width. If you specify G once, spaces are added at that point in the line. If you specify G more than once, spaces are distributed as evenly as possible at every point where you put a G. For example:

Specification	Result
"Hello there"	Hello there
"'G'Hello there"	 there Hello
"'G'Hello there'G'"	Hello there
"Hello'G'there"	Hello there
"'G'Hello'G'there'G'"	Hello there

G Options

If the text includes an apostrophe ('), right bracket (]), caret (^), or backslash (\) that you want to include in the report, type an apostrophe before the character. For example:

```
>LIST PAYABLES WITH DUE DATE <= 1/1/85 HEADING "JANUARY'S  
PAYMENTS 'LDP'"
```

You can also specify the Q option before the first occurrence of any of these characters. You only need to specify Q once in any heading. For example:

HEADING "'Q'Using], ^, and \ in Headings"

HEX

Use with **COPY** and **T.READ** to display data in hexadecimal format.

Use with [LIST.DIFF](#), [LIST.INTER](#), and [LIST.UNION](#) to list output in hexadecimal format.

–HEX

Use with [CP](#) and [CT](#) to copy data in hexadecimal format.

HOLD

Use with [SETPTR \(UNIX\)](#) or [SETPTR \(Windows Platforms\)](#) to send print jobs to the spool queue in the hold state. The jobs are not printed initially but can be printed using [SP.EDIT \(UNIX\)](#), [SP.EDIT \(Windows Platforms\)](#), or [usm -r](#). The jobs can be removed from the spool queue using [SP.EDIT](#) or [usm -k](#). Unlike the RETAIN option, HOLD removes a job from the spool queue once it is printed. The HOLD option is the same as the S option of [SP.ASSIGN](#).

Use with the [SPOOL \(Windows Platforms\)](#) to suspend a print job.

HUSH

Use with [COMO](#) to suppress terminal display.

Use with [GET.TERM.TYPE](#) and [SET.TERM.TYPE](#) to suppress terminal output.

ID.ONLY

Use in a Retrieve sentence or SQL SELECT statement to display only record IDs when the fields in the @ phrase or @LPTR phrase would otherwise be displayed. However, if you use this keyword in a sentence that specifies field names for display, it is ignored. To display only record IDs instead of the fields in the @LPTR or @ phrase, use a sentence like the following:

```
>LIST PAYABLES ID.ONLY
```

In NLS mode, use with the [UNICODE.FILE](#) command to check only the record IDs for data loss.

ID-SUP

Synonym for [ID.SUP](#).

ID-SUPP

Synonym for ID.SUP.

ID.SUP

Use in a Retrieve sentence, a SQL SELECT statement, or a [COPY](#) command to suppress record IDs. Without this keyword, Retrieve displays record IDs in the first column of a report. For example:

```
>LIST PAYABLES WITH BAL.DUE GT 10 ID.SUP
```

IN

Use in any Retrieve sentence to make it more English-like. This keyword is not required and is ignored by Retrieve.

IN2

Use with [UPDATE.ACCOUNT](#) to change the account flavor to IN2.

IN2.FORMAT

Forces IN2-compatible format.

INFO

Use with [CONVERT.SQL](#) to list D-, A-, and S-descriptors in the specified file's dictionary.

INFORM

Use with [SETPTR \(UNIX\)](#) or [SETPTR \(Windows Platforms\)](#) to display spooler job numbers of newly queued jobs. The default is that INFORM is turned off.

INFORMATION

Use with [UPDATE.ACCOUNT](#) to change the account flavor to INFORMATION.

INFORMATION.FORMAT

Use with [T.DUMP](#), [T.LOAD](#), and [T.WTLBL](#) to specify a tape label in a format compatible with Prime INFORMATION. Prime INFORMATION systems do not write tape labels.

INODE

Use with [UNLOCK](#) to restrict lock removal to a file or device specified by the i-node number.

INPLACE

Use with [RESIZE](#) to resize a file in place. This is very useful when free disk space is low. To resize the file PAYABLES in its present disk location, enter the following:

```
>RESIZE PAYABLES 2 3 1 INPLACE
```

On System V systems, INPLACE moves records out of overflow blocks where possible, but does not reduce the size of the UNIX file. In some cases, it actually increases the size of the UNIX file.

INQUIRING

Use in any Retrieve sentence to prompt for the record IDs. Enter a record ID at the Records prompt. To end the prompting, press Return. For example:

```
>LIST PAYABLES INQUIRING
```

INTERNAL

Use with [DEFINE.DF](#) to specify the partitioning algorithm for a distributed file.

INTERSECT

Use with **MERGE.LIST** to produce a select list whose elements are contained in *list1* and *list2*.

INVERT

Use with **CONNECT** to control case inversion for alphabetic characters typed while **CONNECT** is running.

INVISIBLE

Use in any Retrieve sentence to make it more English-like. This keyword is not required and is ignored by Retrieve.

IS.NULL

A relational operator used in selection expressions, WHEN clauses, and IF statements. Its syntax is as follows:

$$\{ \text{WITH} \mid \text{WHEN} \mid \text{IF} \} \text{ field IS.NULL}$$

IS.NULL selects all records containing the null value in *field*.

You must use the IS.NULL keyword to specify the null value in a selection expression. You cannot use the “equal to” operators, since a null value is not equal to anything, including itself.

For example:

```
>SORT INVENTORY WITH PRICE IS.NULL
```

IS.NOT.NULL

A relational operator used in selection expressions, WHEN clauses, and IF statements. Its syntax is as follows:

$$\{ \text{WITH} \mid \text{WHEN} \mid \text{IF} \} \text{ field IS.NOT.NULL}$$

IS.NOT.NULL selects all records that do not contain the null value in *field*.

You must use the IS.NOT.NULL keyword to specify nonnull values in a selection expression. You cannot use the “not equal to” operators, since a null value is neither equal to nor not equal to anything, including itself.

For example:

```
>SORT INVENTORY WITH PRICE IS.NOT.NULL
```

ISOLATION

Use with [SET.SQL](#) to set the isolation level for the current session.

JOIN.BUFFER

Use with [SET.SQL](#) to specify the size of the in-memory join buffer.

KEEP

Use with [SETPTR \(UNIX\)](#) or [SETPTR \(Windows Platforms\)](#) to keep output files open after reports are sent to them. Use in mode 3 to append subsequent reports to the same file.

KEEP.COMMON

Use with [RUN](#) and [RAID](#) to maintain the value of unnamed common variables across BASIC programs connected with the CHAIN statement.

–L

Synonym for –LIST.

LARGE.RECORD

Use with [CREATE.FILE](#), [CONFIGURE.FILE](#), and [RESIZE](#) to specify the size of a record ID considered too large to be included in the primary group buffer of a dynamic (type 30) file. Its syntax is as follows:

```
LARGE.RECORD n
```

n can be an integer or a percentage. Specified as an integer, *n* is the number of bytes a record must contain to be considered a large record. Specified as a percentage, *n* is a percentage of the group size. When the size of a record exceeds the value specified, the data for the record is placed in an overflow buffer, but the record ID is placed in the primary buffer. This method of large record storage increases access speed. The default LARGE.RECORD size is 80%.

LAYER.STACK

Use with [PORT.STATUS](#) to generate a process layer stack dump for the specified job.

LE

The relational operator LESS THAN OR EQUAL TO used in selection expressions, WHEN clauses, and IF statements. For example:

```
>SELECT PAYABLES WITH DUE.DATE LE 12/1/94
```

LENGTH

Use with [SET.TERM.TYPE](#) to set the length of the terminal screen.

LESS

Synonym for [LT](#).

LIKE

Synonym for [MATCHING](#).

LIST

Use with [COMO](#) to list COMO records in your account.

–LIST

Use with [BASIC](#), [FORMAT](#), and [FANCY.FORMAT](#) to generate a listing of the program.

Use with [SPOOL](#) to display a listing of the spool queue.

LNUM

Use with [SETPTR \(UNIX\)](#) or [SETPTR \(Windows Platforms\)](#) to prefix each line with its line number.

LOCAL

Use with [ACCOUNT.FILE.STATS](#) and [LIST.FILE.STATS](#) to use the STAT.FILE file in the local account.

Use with [CATALOG](#) to specify that a BASIC program should be cataloged in the current account rather than in the system catalog. To catalog the subroutine OVERDUE locally, enter the following:

```
>CATALOG BP OVERDUE LOCAL
```

Use with [VCATALOG](#) to verify a locally cataloged program.

LOCK.HIST

Use with [PORT.STATUS](#) to list concurrency control operations.

LOCK.WAIT

Use with [SET.SQL](#) to specify the number of seconds to wait on a record or file lock before returning an error.

LOCKS

Can be included in the [MASTER](#) command for compatibility with Prime INFORMATION.

LPTR

Use in a Retrieve sentence or SQL SELECT statement to send the results to a printer channel instead of the terminal. Its syntax is as follows:

LPTR [*n*]

n is an integer, 0 through 255. The default is 0. Printer channels are assigned with the [SETPTR \(UNIX\)](#) or [SETPTR \(Windows Platforms\)](#) command.

For example:

>LIST PAYABLES PYMT.DUE LPTR 3

Use with [ASSIGN](#) to specify a logical print channel.

LT

The relational operator LESS THAN used in selection expressions, WHEN clauses, and IF statements. For example:

>SELECT PAYABLES WITH DUE.DATE LT 12/1/94

MAP

In NLS mode, use in the [ASSIGN](#), [T.ATT](#), or [SET.TERM.TYPE](#) command to set a map for the tape device or terminal.

MARGIN

Use in a Retrieve sentence or SQL SELECT statement to define the width of the left margin by indenting from the left edge. If you do not use this keyword, the display is left-justified, with a left margin of 0. To change the left margin in a report, use a sentence like this:

**>LIST PAYABLES WITH DUE.DATE <= 1/1/95 HEADER
"JANUARY PAYMENTS 'LDP'" MARGIN 5**

MATCHES

Synonym for MATCHING.

MATCHING

A relational operator used in selection expressions, WHEN clauses, IF statements, and ReVise sentences.

Use in selection expressions, WHEN clauses, and IF statements to compare a field value or prompt response to the pattern you specify. The match is successful if the value matches the pattern. The pattern can contain any combination of letters, numbers, or the special characters X, A, and N preceded by an integer used as a repeating factor. For example, 8N is the pattern for character strings with eight numeric characters.

The following table lists the pattern codes and their definitions:

Pattern	Definition
...	Any number of any characters (including none).
0X	Any number of any characters (including none).
<i>n</i> X	<i>n</i> number of any characters.
0A	Any number of alphabetic characters (including none).
<i>n</i> A	<i>n</i> number of alphabetic characters.
0N	Any number of numeric characters (including none).

MATCHING Pattern Codes

Pattern	Definition
<i>n</i> N	<i>n</i> number of numeric characters.
' <i>text</i> '	Exact text; any literal string (quotation marks required).
" <i>text</i> "	Exact text; any literal string (quotation marks required).

MATCHING Pattern Codes (Continued)

If *n* is longer than nine digits, it is used as text in a pattern rather than as a repeating factor for a special character. For example, the pattern 1234567890N is treated as a literal string, not as a pattern of 1,234,567,890 numeric characters.

You can specify a match on a string at the beginning, end, or anywhere in the field. If the pattern contains any special characters (such as * or –) or a blank space, enclose the entire pattern in quotation marks and the string in another set of quotation marks. The pattern has one of the following syntaxes:

Syntax	Description
<i>string</i> ...	Matches records with a field beginning with <i>string</i> .
... <i>string</i> ...	Matches records with <i>string</i> anywhere in the field.
... <i>string</i>	Matches records with a field ending with <i>string</i> .
<i>string1</i> ... <i>string2</i>	Matches records with a field beginning with <i>string1</i> , ending with <i>string2</i> , and containing any number of characters between them.

Pattern Matching Syntax

The three dots are part of the syntax and must be typed.

In a ReVise sentence MATCHING tells ReVise to accept only information matching the pattern as a new value for the specified field. Its syntax is as follows:

field { MATCHING | MATCHES | LIKE } *pattern*

field is one of the field names in the ReVise sentence. ReVise prompts only for the specified field unless you specify additional fields in the ReVise sentence. You can use more than one MATCHING clause in a ReVise sentence.

To create a select list with zip codes beginning with 01, enter the following:

>SELECT MAIL WITH ZIP MATCHING 01...

To create a report showing only information for vendors with AC in the vendor code, enter the following:

>LIST PAYABLES WHEN CODE MATCHING ...AC...

To execute a different sentence for all entries with zip codes ending in 67, use an IF statement such as the following in a paragraph:

IF <<ZIP>> MATCHING ...67 THEN GO ZIP67

In the following example, the date you enter into the MEMBERSHIP file must begin with FEB:

>REVISE MEMBERSHIP DATE MATCHING FEB...

MAX

Use in a Retrieve sentence to calculate and list the maximum value of a field in a set of records. Its syntax is as follows:

MAX *field.expression* [NO.NULLS]

field.expression specifies the name of a field or an EVAL expression for which you want to display the maximum value.

NO.NULLS tells Retrieve to ignore empty string values.

The maximum value is the value that appears first when you use a Retrieve sort by descending order on the field. This is based on the field's justification in the dictionary or the justification specified by the **FMT** keyword. This means that you can use the MAX keyword on alphabetic field values as well as numeric values.

When you use NO.NULLS in the sentence, Retrieve ignores field values that contain empty string values. If you use NO.NULLS and all the fields you select have empty string values, an empty string value is listed as the maximum.

The output format depends on the justification defined in the dictionary definition of the field. If the field uses the default column header, the MAX keyword is added before the column header.

If you specify a multivalued field, Retrieve treats each value separately.

If you use the MAX keyword with a breakpoint keyword ([BREAK.ON](#) or [BREAK.SUP](#)), Retrieve displays the field values for each record on the detail lines and the maximum breakpoint value for each breakpoint in addition to the overall maximum value at the bottom of the report.

For example, to display each salary in the records of the PERSONNEL file with the maximum salary at the bottom of the report, enter the following:

>LIST PERSONNEL LAST.NAME MAX SALARY

ME

Use with [MESSAGE](#) to display the status of receive mode for all users with your user name.

Use with [STATUS](#) to display information about your current processes.

MERGE.LOAD

Use with [CREATE.FILE](#), [CONFIGURE.FILE](#), and [RESIZE](#) to specify the level at which a dynamic (type 30) file's modulo is decreased by 1. Its syntax is as follows:

MERGE.LOAD *n*

n is a number indicating a percentage of the space allocated for the file. The default MERGE.LOAD is 50%. When the data in the file is less than the percentage, the data in the last group of the file is merged with another group, to decrease the modulo by 1.

MFILE.HIST

Use with [PORT.STATUS](#) to list the most recent log of rotating file pool (MFILE) operations.

MIN

Use in a Retrieve sentence to calculate and list the minimum value of a field in a set of records. Its syntax is as follows:

MIN *field.expression* [NO.NULLS]

field.expression specifies the name of a field or an EVAL expression for which you want to list the minimum value. RetrieveVe treats nonnumerics as zero values.

NO.NULLS tells RetrieveVe to ignore empty string values.

The minimum value is the same as the first value that appears when you use a RetrieveVe sort on the field depending on the specified justification. This means that you can use the MIN keyword on alphabetic field values, as well as numeric values.

When you use the NO.NULLS keyword, RetrieveVe ignores fields containing an empty string value. However, if you use NO.NULLS and all the fields you select have empty string values, an empty string value is listed as the minimum.

The output format of the display depends on the justification specified in the file dictionary. If the field uses the default column header, MIN is added before the column header.

If you specify a multivalued field, RetrieveVe treats each value separately.

If you use the MIN keyword with a breakpoint keyword (**BREAK.ON** or **BREAK.SUP**), RetrieveVe displays the field values for each record on detail lines with the minimum breakpoint value for each breakpoint in addition to the overall minimum value at the bottom of the report.

For example, to display each salary in the records of the PERSONNEL file with the minimum salary at the bottom of the report, excluding empty string values, enter the following:

```
>LIST PERSONNEL LAST.NAME MIN SALARY NO.NULLS
```

MINIMIZE.SPACE

Use with **CREATE.FILE**, **CONFIGURE.FILE**, and **RESIZE** to specify that the values for split load, merge load, and the large record size be calculated to optimize the amount of space required by a dynamic (type 30) file at the expense of access time. If you specify a value for split load, merge load, or large record size, the value you specify overrides the value calculated by MINIMIZE.SPACE. If MINIMIZE.SPACE and **RECORD.SIZE** are specified, the value for the large record size calculated by MINIMIZE.SPACE is used instead of the value calculated by RECORD.SIZE.

MINIMUM.MODULUS

Use with [CREATE.FILE](#), [CONFIGURE.FILE](#), and [RESIZE](#) to specify the minimum modulo of a dynamic (type 30) file. Its syntax is as follows:

MINIMUM.MODULUS *n*

n is an integer of 1 (the default) or greater. This value is also the initial value of the modulo of the dynamic file.

–MODIFY

Use with [SPOOL](#) to change the attributes of a print job already queued for printing.

MONETARY

Use with [SET.LOCALE](#) to set the MONETARY category.

MTU

Specifies the magnetic tape unit and needs to be used only if you have more than one tape drive or if you want to change the mode. Its syntax is as follows:

MTU [*mtu*]

If you do not use the optional *mtu* qualifier, the default 000 is used.

Letter	Meaning	Possible Values
<i>m</i>	Mode	0 – ASCII, no conversions 1 – EBCDIC conversion 2 – Invert high-order bit 3 – Invert high-order bit and EBCDIC conversion
<i>t</i>	Tracks	Not used
<i>u</i>	Unit number	Any number from 0 through 7 indicating the tape drive number

mtu Qualifiers

Use MTU with [T.DUMP](#) and [T.LOAD](#), the commands to create and restore magnetic tape records. To use MTU with T.DUMP, enter a command similar to this:

```
>T.DUMP PAYABLES MTU 001
```

If you create tapes on a UniVerse system that are to be used on a Prime system, or vice versa, specify *mtu* as 200 in order to invert the high order bit.

Use MTU with [PTERM \(UNIX\)](#) or [PTERM \(Windows Platforms\)](#) to specify terminal characteristics of a magnetic tape channel.

MULTI.VALUE

Use as a field qualifier in a Retrieve sentence or SQL SELECT statement to specify that a field name or EVAL expression be treated as multivalued. The MULTI.VALUE keyword always follows the name of the field or EVAL expression. MULTI.VALUE overrides any existing specification in field 6 of the file dictionary. See also the [SINGLE.VALUE](#) keyword.

You cannot use MULTI.VALUE in the same field expression with the SINGLE.VALUE field qualifier.

MVDISPLAY

Use with [CONNECT](#) to define how to display value marks in multivalued data (when connected to a UniVerse data source).

NE

The relational operator NOT EQUAL used in selection expressions, WHEN clauses, and IF statements. For example:

```
>SORT INVENTORY WITH ITEM NE 'SHIRTS'
```

NEEDNL

Use with [SET.TERM.TYPE](#) to let UniVerse generate newline sequences.

NETWORK

Use with **STATUS** to display node information about the system.

NEW.PAGE

Use with the **COPY** command to list each record on a separate page.

NEWACC

Use with **UPDATE.ACCOUNT** to change the account flavor to IDEAL.

NEXT

Use with the **BANNER** keyword in the **SETPTR (UNIX)** or **SETPTR (Windows Platforms)** commands to specify a unique record ID in the **&HOLD&** file for each new report.

NEXT.AVAILABLE

Use in a ReVise sentence to tell ReVise to use the next sequential value for the record ID every time you press Return at the **RECORD ID=** prompt.

When you first use the **NEXT.AVAILABLE** keyword, ReVise creates the **&NEXT.AVAILABLE&** record in the dictionary and gives it the value 1. This is an X-type dictionary record. You can change the next available number at any time using either ReVise or ED. You can put the **NEXT.AVAILABLE** keyword in the **@REVISE** phrase.

Every time you use **NEXT.AVAILABLE**, ReVise automatically adds 1 to the value in the **&NEXT.AVAILABLE&** record. ReVise then verifies that the value does not exist already. If it does, ReVise continues to add 1 to the value until it comes to an unused value and assigns it as the record ID.

If you enter a value instead of pressing Return, ReVise uses the value you enter.

ReVise uses the next sequential ID when you press Return to enter data in the INVENTORY file:

>REVISE INVENTORY NEXT.AVAILABLE

NFMT

Use with [SETPTR \(UNIX\)](#) or [SETPTR \(Windows Platforms\)](#) to allow the application to control pagination and formatting instead of the spooler.

NHEAD

Use with [SETPTR \(UNIX\)](#) or [SETPTR \(Windows Platforms\)](#) and [SPOOL](#) to suppress printing of the banner.

NO

Synonym for [NE](#) (not equal).

Use with [ENABLE.RECOVERY](#) to remove the current logging into file.

NO.INDEX

Use in a Retrieve command to specify that secondary key indexes not be used if they exist. This is useful when indexes are not up to date or built.

NO.MATCH

Use with [SEARCH](#) to specify that only records not containing any of the specified strings should be selected.

NO.NEW

Use with [COPY](#) to copy a record only if a record with the same record ID exists in the target file. The new record overwrites the old.

NO.NULLS

Use with [CREATE.INDEX](#) to specify that empty secondary key values are not to be indexed in the newly created indexes. You can save disk space and processing time using this facility, but reports generated from empty-string-suppressed secondary keys do not contain values for the empty keys.

Use with the [AVG](#), [ENUMERATE](#), [MAX](#), and [MIN](#) aggregate function keywords, and with the [SAVING](#) clause of the [SELECT](#) and [SSELECT](#) commands, to suppress empty string values.

NO.PAGE

Synonym for [NOPAGE](#).

NO.SELECT

Use with [SEARCH](#) to specify that a list of the record IDs be displayed instead of a select list being created.

NO.SPLIT

Use with [LIST](#) or [SORT](#) to start a record on a new page if it does not fit on the current page of a report.

NO.WAIT

Use with [LOCK](#) to return control to the command processor if the specified lock is already set. If this option is not used, [LOCK](#) waits for the other user to release the lock. To return control to the command level if [LOCK](#) number 42 is set, use a sentence like the following:

>LOCK 42 NO.WAIT

NO.WARN

Use with [RUN](#) and [RAID](#) to suppress all warning (nonfatal) error messages. If NO.WARN is not specified, the terminal screen prints run-time error messages as they are encountered.

Use with [LIST.DIFF](#), [LIST.INTER](#), and [LIST.UNION](#) to suppress line numbers.

NODE

Use with [UNLOCK](#) to restrict lock removal to the network node specified by its argument.

NODEFAULT

Use with [SETPTR \(UNIX\)](#) or [SETPTR \(Windows Platforms\)](#) to change only those parameters specified with your command and not supply default settings for unspecified parameters.

NOEJECT

Use with [SETPTR \(UNIX\)](#) or [SETPTR \(Windows Platforms\)](#) to indicate not to skip to a new page at the end of a report.

NOFMT

Synonym for [NFMT](#).

NOHEAD

Synonym for [NHEAD](#).

–NOHEAD

Use with [SPOOL](#) to suppress the flag page.

NOHOLD

Use with [SETPTR \(UNIX\)](#) or [SETPTR \(Windows Platforms\)](#) and [SPOOL](#) to turn off the HOLD option.

Use with the [SPOOL \(Windows Platforms\)](#) command to resume a suspended print job.

NOKEEP

Use with [SETPTR \(UNIX\)](#) or [SETPTR \(Windows Platforms\)](#) to close output files after reports are sent to them.

NONE

In NLS mode, use with most NLS commands, such as the [SET.FILE.MAP](#) command, to specify that no mapping takes place. Records are read and written to the file in internal format.

NOPAGE

Use in a Retrieve sentence or SQL SELECT statement to scroll the output on the screen and not stop at page breaks. Without NOPAGE, Retrieve displays a page of information and the following message:

Press any key to continue...

To display the next screen, press any key. To return to the UniVerse prompt without displaying the next screen, press Q or Ctrl-X. The page size is set to 24 lines per screen, although you can change this value using the [TERM](#) command.

For example:

```
>LIST PAYABLES PYMT.DATE NOPAGE
```

NORETAIN

Use with [SETPTR \(UNIX\)](#) or [SETPTR \(Windows Platforms\)](#) to turn off the RETAIN option.

NOT

Synonym for [NE](#).

NOT.MATCHING

A relational operator used in selection expressions, WHEN clauses, IF statements, and ReVise sentences. NOT.MATCHING is the inverse of MATCHING.

Use in selection expressions, WHEN clauses, and IF statements to compare a field value or prompt response to the pattern you specify. The match is successful only if the value does not match the pattern. The pattern can contain any combination of letters, numbers, or the special characters X, A, and N preceded by an integer used as a repeating factor. For example, 8N is the pattern for character strings with eight numeric characters.

The following table lists the pattern codes and their definitions:

Pattern	Definition
...	Any number of any characters (including none).
0X	Any number of any characters (including none).
<i>n</i> X	<i>n</i> number of any characters.
0A	Any number of alphabetic characters (including none).
<i>n</i> A	<i>n</i> number of alphabetic characters.
0N	Any number of numeric characters (including none).
<i>n</i> N	<i>n</i> number of numeric characters.
' <i>text</i> '	Exact text; any literal string (quotation marks required).
" <i>text</i> "	Exact text; any literal string (quotation marks required).

NOT.MATCHING Pattern Codes

If *n* is longer than nine digits, it is used as text in a pattern rather than as a repeating factor for a special character. For example, the pattern 1234567890N is treated as a literal string, not as a pattern of 1,234,567,890 numeric characters.

If the pattern contains any special characters (such as * or –) or a blank space, enclose the entire pattern in quotation marks and the string in another set of quotation marks. The pattern has one of the following syntaxes:

Syntax	Definition
<i>string</i> ...	Matches records with a field beginning with <i>string</i> .
... <i>string</i> ...	Matches records with <i>string</i> anywhere in the field.
... <i>string</i>	Matches records with a field ending with <i>string</i> .
<i>string1</i> ... <i>string2</i>	Matches records with a field beginning with <i>string1</i> , ending with <i>string2</i> , and with any number of characters between them.

NOT.MATCHING Pattern Syntax

The three dots are part of the syntax and must be typed.

Used in a ReVise sentence, NOT.MATCHING tells ReVise to accept only information that does not match the pattern as a new value for the specified field. Its syntax is as follows:

field { NOT.MATCHING | UNLIKE } *pattern*

field is one of the field names in the ReVise sentence. ReVise prompts only for the fields whose names do not match *pattern* unless you specify additional fields in the ReVise sentence. You can use more than one NOT.MATCHING clause in a ReVise sentence.

To create a select list that does not include zip codes beginning with 11, enter the following:

>SELECT MAIL WITH ZIP NOT.MATCHING 11...

To create a report that does not include information for vendors with AC in the vendor code, enter the following:

>LIST PAYABLES WHEN CODE NOT.MATCHING ...AC...

To execute a different sentence for all entries that do not have a zip code ending in 42, use the following IF statement in a paragraph:

IF <<ZIP>> NOT.MATCHING ...42 THEN GO NOT42

NOXREF

Use with [CATALOG](#) to catalog the program without the cross-reference table and symbol table information. To catalog the program UPDATE.FILES without its cross-reference and symbol tables, enter the following:

```
>CATALOG BP UPDATE.FILES NOXREF
```

NULL

Use with [CONNECT](#) to define how to display the SQL null value.

Use with [SET.INDEX](#) to remove the current path for the indexes in the specified file from the file header.

NUM.SUP

Use with [COPY](#) to suppress line numbers.

NUMERIC

Use with [SET.LOCALE](#) to set the NUMERIC category.

ODBC.CONNECTIONS

Use with [PORT.STATUS](#) to list the data source name, the DBMS type, and the network node name for the connection between the UniVerse system and the server.

OF

Use in any Retrieve sentence to make it more English-like. This keyword is not required and is ignored by Retrieve.

OFF

Use with commands such as [BREAK](#) or [DIVERT.OUT](#) to stop or disable a process or function.

Use with [MASTER](#) to log out users.

ON

Use with commands such as [BREAK](#) or [DIVERT.OUT](#) to begin or enable a process or function.

Use with [SET.REMOTE.ID](#) to specify the name of a remote system running UniVerse.

ONLY

Synonym for [ID.ONLY](#).

OPTIM.SCAN

Use with [SET.SQL](#) to turn optimistic scanning on or off.

OR

Logical operator OR used to join selection expressions, WHEN clauses, and IF statements. For example:

>LIST MAIL WITH STATE EQ MA OR INTEREST EQ PAINT

OS

Use with [SET.GCI.MAP](#) to set the GCI map to the operating system map specified by the NLSDEFOSMAP configurable parameter.

OVERWRITING

Use with commands such as **T.LOAD** or **COPY** to overwrite records that exist in the target file. If you do not use this keyword in your sentence, T.LOAD or COPY examines the target file and does not copy a record if a record with the same record ID already exists. For example:

>T.LOAD PAYABLES OVERWRITING

Use with **LIST.DIFF**, **LIST.INTER**, and **LIST.UNION** to overwrite an existing list in the **&SAVEDLISTS&** file.

Use with **SETFILE** to overwrite an existing file pointer in the VOC file.

PCT

Synonym for **PERCENT**.

PDICT

Use with a file name to specify a Pick-style file dictionary rather than the data file. PDICT causes the Pick dictionary, as specified by the *P_filename* in field 5 of the VOC entry, to be displayed using the **DICT.PICK** file rather than the **DICT.DICT** file. Specify PDICT immediately before the file name. For example:

>REVISE PDICT PAYABLES

>LIST PDICT PAYABLES WITH TYPE EQ PH

PERCENT

Use in a Retrieve sentence or SQL SELECT statement to calculate percentages for a numeric field. It calculates the total value of the specified field for all records, then calculates and displays the percent of the total value of the specified field for each record. Its syntax is as follows:

PERCENT ['*n*'] *field*

The option *n* is an integer from 0 through 5 that must be enclosed in single quotation marks. This integer specifies the number of digits to be displayed after the decimal point. If you do not use this option, two digits are displayed.

field is the name of the field you want to find percentages for.

For example:

>LIST INVENTORY TOTAL GAMES PERCENT GAMES

PERCENT.GROWTH

Use with [HASH.HELP](#) to change allowance for file growth.

PERCENTAGE

Synonym for PERCENT.

PICK

Use with [HELP](#) to list help for PICK, REALITY, and IN2 flavor accounts.

Use with [UPDATE.ACCOUNT](#) to change the account flavor to PICK.

PICK.FORMAT

Use with [T.DUMP](#), [T.LOAD](#), and [T.WTLBL](#) to specify a tape label in a format compatible with Pick systems. The label includes the time and date. With the T.WTLBL command text can follow the PICK.FORMAT keyword to be included in the label.

Pick labels (including ADDS Mentor and Ultimate) are 80 bytes long and have the following format ("" represents an ASCII blank, **I** represents an item mark, and **F** represents a field mark):

```
I L p xxxx "" HH:MM:SS
"" DD "" MMM "" YYYY
"" filename "" header pad F rr
```

I	Item mark (segment mark)
L	ASCII character L
<i>xxxx</i>	Tape record size (hexadecimal)
<i>HH:MM:SS</i>	Time (external 24-hour)
<i>DD MMM YYYY</i>	Date (external form)
<i>filename</i>	Name of the file dumped
<i>header</i>	(Optional) Label header text
<i>pad</i>	Enough blanks to get to byte 78
F	Field mark (attribute mark)
<i>rr</i>	Reel number (hexadecimal)

PID

Use with [PORT.STATUS](#) to limit the report to a specific process.

PIOPEN

Use with [UPDATE.ACCOUNT](#) to change the account flavor to PIOPEN.

PORT

Use with [PORT.STATUS](#) to limit the report to jobs running on a specific port.

-PORTNO

Use with [SPOOL](#) to specify the port number of the user who submitted the print job.

PREFIX

Use with [CONNECT](#) to define the prefix character for local commands.

PRINT

Use in any Retrieve sentence to make it more English-like. This keyword is not required and is ignored by Retrieve.

PRINTER

Synonym for [AT](#).

–PRINTER

Use with [SPOOL](#) to specify the name of a printer.

PRIORITY

Use with the [SETPTR \(UNIX\)](#), [SETPTR \(Windows Platforms\)](#), and [SPOOL \(Windows Platforms\)](#) commands to specify priority for printing. The highest priority is 1 and the lowest is 255. The priority you enter is converted to a number within the range for Windows platforms, which goes from 99 through 1. Because the range of priorities in UniVerse goes from 1 through 254, the mapping is not one to one. The following list shows the conversion of UniVerse priorities to Windows priorities:

Priority in UniVerse	Priority in Windows

1	99
2	98
.	.
.	.
.	.
10	90
11	89
12	89
13	89
14	88
15	88
16	88
17	87
.	.
.	.
.	.
244	11
245	10
246	9
247	8
.	.
.	.
.	.
254	1

PROGRAMSIZE

Use with **LIMIT** to set the total size of user memory space for storing active BASIC routines. When a routine exceeding the size specified with PROGRAMSIZE is loaded into user memory space, UniVerse tries to unload the least recently used programs to bring the total usage within the limit. Size is specified as a number of 1024-byte units. A value of 0 specifies unlimited space.

PROMPT

Use with **LOCK** to return control to the command processor so that a user can execute other commands. When the lock becomes free, it is set and a message is issued indicating that the lock is now set. You are not notified until the next prompt unless you specify **NOTIFY ON**.

Use with [SET.REMOTE.ID](#) to prompt users to enter their passwords.

–RANGE

Use with [SPOOL](#) and [SPOOL \(Windows Platforms\)](#) to specify a range of print jobs.

READLLOCK

Use with [UNLOCK](#) to remove only shared record locks.

READULOCK

Use with [UNLOCK](#) to remove only update record locks.

REALITY

Use with [UPDATE.ACCOUNT](#) to change the account flavor to REALITY.

REALITY.FORMAT

Use with [T.DUMP](#), [T.LOAD](#), and [T.WTLBL](#) to specify a tape label in a format compatible with REALITY systems. The label includes the time and date. With the T.WTLBL command, text can follow the REALITY.FORMAT keyword to be included in the label.

REALITY labels are 78 bytes long and have the following format ("" represents an ASCII blank, **␣** represents an item mark, **F** represents a field mark, and **v** represents a value mark):

```
␣ L header v HH:MM:SS
"" DD "" MMM "" YYYY
F rr F xxxxx F ␣ pad
```

␣	Item mark (segment mark)
L	ASCII character L
<i>header</i>	(Optional) Label header text (<= 44 characters)
v	Value mark (follows L if no header)
<i>HH:MM:SS</i>	Time (external 24-hour)
<i>DD MMM YYYY</i>	Date (external form)
F	Field mark (attribute mark)
<i>rr</i>	Reel number (decimal)
<i>xxxxx</i>	Tape record size (decimal)
<i>pad</i>	Fill blanks up to 78 bytes

RECORD

Use with [UNLOCK](#) to restrict lock removal to a specific record.

RECORD.SIZE

Use with [CREATE.FILE](#), [CONFIGURE.FILE](#), and [RESIZE](#) to specify that the values for group size, and large record size for a dynamic (type 30) file should be calculated based on the value of the estimated average record size specified. Its syntax is as follows:

```
RECORD.SIZE n
```

n is your estimate of the average record size for the dynamic file, specified in bytes. RECORD.SIZE does not limit the size of records. If you specify a value for group size, or large record size, the value you specify overrides the value calculated by RECORD.SIZE.

–REJECT

Use with [MESSAGE](#) to disable receive mode for all messages.

REMOVING

Use with **DEFINE.DF** to remove a part file from a distributed file.

REPORTING

Use with **COPY** to display the status of the **OVERWRITE**, **UPDATING**, and **DELETING** options, and record IDs of the source records and the record IDs of the copies.

REQUEUE

Synonym for **RETAIN**.

REQUIRE.INDEX

Use in a Retrieve command to specify that secondary key indexes must be used to process the sentence. If indexes cannot be used, an error message appears and the sentence does not proceed.

REQUIRE.SELECT

Use with any Retrieve sentence when you want to process a select list. This keyword requires an active select list in order to run the process. If a select list is not active, a message like the following appears:

No active list found, processing terminated.

If that happens, use **SELECT**, **SSELECT**, **QSELECT**, **FORM.LIST**, or **GET.LIST** to make a select list active, then repeat the selection sentence.

For example:

>LIST PAYABLE REQUIRE.SELECT

RESTORE

Use with [CONVERT.SQL](#) to restore a previously converted table to a UniVerse file, leaving the data in its current state.

RESTOREDATA

Use with [CONVERT.SQL](#) to restore a previously converted table to a UniVerse file, restoring both the data file and the dictionary to their original contents.

RETAIN

Use with [DEFINE.DF](#) to retain a part file's number and algorithm in the part file after it is removed from a distributed file.

Use with [SETPTR \(UNIX\)](#) or [SETPTR \(Windows Platforms\)](#) to hold jobs in the spool queue after printing them. Held jobs can be reprinted using [usm](#) with the *-r* option or by using the [SP.EDIT \(UNIX\)](#) or [SP.EDIT \(Windows Platforms\)](#) command. Reprinting the job does not remove it from the spool queue. Held jobs can be removed from the spool queue using the *-k* option of *usm*, or by using the SP.EDIT command. The RETAIN option is the same as the H option of SP.ASSIGN.

RUNNING

Use with [PORT.STATUS](#) to list all unsuspended UniVerse processes.

SAID

A relational operator used in WITH expressions, WHEN clauses, and IF statements to select values that sound like the specified value. The specified value must begin with the same letter as the value. To create a list of all names like Paine, Payne, Pane, and so on, enter the following:

```
>LIST PAYABLES WITH LNAME SAID PAIN
```

SAMPLE

Use in a Retrieve command to limit the number of records selected. Its syntax is as follows:

SAMPLE [*n*]

n is a number specifying the upper limit on the number of records selected. The first *n* records are selected. It is useful to list part of a long report in order to check the formatting. To list the first five records in the INVENTORY file, enter the following:

>LIST INVENTORY SAMPLE 5

SAMPLED

Use in a Retrieve command to limit the number of records selected. SAMPLED selects every *n*th record, whereas SAMPLE selects the first *n* records. Its syntax is as follows:

SAMPLED [*n*]

n is the number specifying how often to select a record.

SAVEDATA

Use with [CONVERT.SQL](#) to save the contents of the original UniVerse data file.

SAVING

Use with [SELECT](#), [SSELECT](#), or [QSELECT](#) to create a list containing field values instead of record IDs. If you have field values that are also record IDs in another file, this is useful. The syntax in SELECT or SSELECT sentences is as follows:

SAVING [UNIQUE] *field*

To create a list of names using the ACCOUNTS file based on the amounts due in the PAYABLES file, you could use a sentence like the following:

>SELECT PAYABLES WITH AMT.DUE > 1 SAVING NAMES

The UNIQUE keyword omits duplicate values in the saved field from the select list.

The syntax in a QSELECT sentence is as follows:

SAVING *n*

n is the number of the field whose values are to make up the select list.

If you specify multiple SAVING clauses in a SELECT or SSELECT sentence, the entire value of each field becomes an element in the select list. Value marks separate multiple values, subvalue marks separate subvalues.

If you specify multiple SAVING clauses in a QSELECT sentence, each value and subvalue becomes an element in the select list.

SCHEMA

Use with [VERIFY.SQL](#) to verify data about a schema and the SICAs of all its tables and views against data in the SQL catalog.

SCHEMAS

Use with [VERIFY.SQL](#) to verify data about all schemas on the system against data in the SQL catalog.

SELECT.BUFFER

Use with [SET.SQL](#) to specify the size of the in-memory select list buffer.

SELECT.ONLY

Synonym for [REQUIRE.SELECT](#).

SEQ.NUM

Use with [CREATE.FILE](#), [CONFIGURE.FILE](#), and [RESIZE](#) to specify the sequential numeric hashing algorithm for dynamic (type 30) files. The sequential numeric hashing algorithm should be used only for files with record IDs that are mainly numeric, sequential, and consecutive. For other files use the GENERAL keyword to specify the general hashing algorithm. GENERAL is the default hashing algorithm for dynamic files.

SET

Use with [ENVIRONMENT](#) or [ENV](#) to set the value of an environment variable.

SHOW

Use with [CONVERT.SQL](#) to display a generated CREATE TABLE statement.

SINGLE.VALUE

Used as a field qualifier in a Retrieve sentence or SQL SELECT statement, SINGLE.VALUE specifies that the field name or EVAL expression it qualifies be treated as singlevalued. The SINGLE.VALUE keyword always follows the name of the field or EVAL expression. SINGLE.VALUE overrides any existing specification in field 6 of the file dictionary. See also the [MULTI.VALUE](#) keyword.

You cannot use SINGLE.VALUE in the same field expression with the MULTI.VALUE, ASSOC, or ASSOC.WITH field qualifiers.

SOME

Use in any Retrieve sentence to make it more English-like. This keyword is not required and is ignored by Retrieve.

SPLIT.LOAD

Use with [CREATE.FILE](#), [CONFIGURE.FILE](#), and [RESIZE](#) to specify the level at which a dynamic (type 30) file's modulo is increased by 1. Its syntax is as follows:

SPLIT.LOAD *n*

n is a number indicating a percentage of the space allocated for the file. The default SPLIT.LOAD is 80%. When the data in the file exceeds the percentage, the data in one of the groups is divided equally between itself and a new group, to increase the modulo by 1.

SPOKEN

Synonym for [SAID](#).

SPOOL

Use with [COMO](#) to print a COMO record or display it on your screen.

–SPOOL

Use with [BASIC](#) to generate a listing of the program, and spool it directly to the printer rather than to a file.

SQL

Use with [HELP](#) to list help about UniVerse SQL statements.

SQUAWK

Use with [COPY](#) to display a message for each record copied, indicating the record ID of the source record, the record ID of the copy, and the status of the OVERWRITE and DELETING options. To copy the file CUR.ACCOUNTS to the file OLD.ACCOUNTS and list each record ID as it is copied, enter the following:

**>COPY FROM CUR.ACCOUNTS TO OLD.ACCOUNTS ALL
SQUAWK**

Use with [PHANTOM](#) to display the record ID of the record created in the [&PH&](#) file by the phantom process. For example:

```
>PHANTOM SQUAWK RUN BP TEST
```

```
RUN_54385_10131 record has been created in the '&PH&' file.
```

Use with [SEARCH](#) to display a message for each record selected.

STARTPAGE

Use with [SETPTR \(UNIX\)](#) or [SETPTR \(Windows Platforms\)](#) to specify starting page number for printing.

STATISTICS

Use with [ANALYZE.FILE](#) to display additional information in the report for each file. The data displayed with the STATISTICS keyword takes additional time to compile. While the data is compiled, a progress report on data compilation is displayed. After the data is compiled, you are asked to press Return to continue. The next page of information contains the information from the standard report, as well as the total number of records, the number of large records, the number of deleted records, the total size of the record data and record IDs, the unused space, and the total space for records. Press ENTER to get the next screen of information.

The third page of the report displays information about records and groups. The average, minimum, maximum, and standard deviation of the number in all groups of disk records, records, large records, deleted records, data bytes, record ID bytes, unused bytes, and total bytes are displayed. The average, minimum, maximum, and standard deviation of the number in all records of data bytes, record ID bytes, unused bytes, and total bytes are displayed.

Use with [LIST.INDEX](#) to specify that additional information about the secondary key index should be displayed, including the total number of keys, records per key, and index size. Each specified index must be scanned to gather the data, so this process may take a while for large files.

STATS

Synonym for STATISTICS.

–STATUS

Use with [MESSAGE](#) to display the status of receive mode for all users.

SUPP

Synonym for [HDR.SUP](#).

–SUPPRESS

Use with [MESSAGE](#) to display messages on one line, suppressing the “Message from” line and the trailing carriage return and linefeed.

SUSPENDED

Use with [PORT.STATUS](#) to list all suspended UniVerse processes.

SYSTEM

Use with [DEFINE.DF](#) to specify the default partitioning algorithm for a distributed file.

TABLE

Use with [VERIFY.SQL](#) to verify the contents of a table’s SICA against data in the SQL catalog.

TAPE

Direct output of [REFORMAT](#) and [SREFORMAT](#) commands to tape.

TEMPL

Synonym for [USING](#).

TEST

Use with [CONVERT.SQL](#) to analyze a file dictionary or the SQLDEF file and list column and association definitions without converting the file to a table.

In NLS mode, use with [UNICODE.FILE](#) to verify that no data loss occurs during the file conversion. If there is data loss, the file is not converted and a report is displayed.

THAN

Use in any RetrieveVe sentence to make it more English-like. This keyword is not required and is ignored by RetrieveVe.

THE

Use in any RetrieveVe sentence to make it more English-like. This keyword is not required and is ignored by RetrieveVe.

THEN

Use in any RetrieveVe sentence to make it more English-like. This keyword is not required and is ignored by RetrieveVe.

TIME

Use with [SET.LOCALE](#) to set the TIME category.

TO

Use with [ASSIGN](#) or [T.ATT](#) to specify the logical device to which you are assigning a physical device.

Use with [CNAME](#) to specify the new name of a file or record.

Use with [T.READ](#) to specify the tape record at which to stop reading.

Use with [SET.INDEX](#) to set or change the pathname of a file's secondary index directory in the file header.

Use with **SELECT**, **SSELECT**, **QSELECT**, **NSELECT**, **GET.LIST**, **FORM.LIST**, **SEARCH**, or **ESEARCH** to specify which select list to write the list to. The syntax for a select expression is as follows:

TO *n*

n is a number from 0 through 10 specifying which select list should be used.

Use in a **COPY** sentence to specify the file to copy records to. In this case the syntax is as follows:

TO *target.file*

target.file is the file you are copying records to.

To select records with a payment date after 2/1/92 to select list 5, enter the following:

>SELECT PAYABLES WITH PYMT.DATE > 2/1/92 TO 5

To copy records from ACCOUNTS to ACCOUNTS.TMP, enter the following:

>COPY FROM ACCOUNTS TO ACCOUNTS.TMP

TOTAL

Use in a Retrieve sentence or SQL SELECT statement to calculate and display totals for numeric fields. This keyword is often used with breakpoints to produce subtotals. The TOTAL keyword is not the same as the TOTAL function used in I-descriptors and with the CALC keyword, although in some cases it can produce the same result. Its syntax is as follows:

TOTAL *field*

field is the name of the field you want to total.

When TOTAL is used in a breakpoint sentence, put TOTAL immediately before the field it is to total. TOTAL displays the subtotal for each value under the row of dashes (- - -) that indicates the breakpoint. If a breakpoint consists of only one line, the subtotal is the same as the detail value.

The grand total for the field is displayed on the last page of the report under a row of equal signs (====).

The following example lists unit price subtotals for each product in the PAYABLES file, and lists a grand total at the end of the report:

```
>LIST PAYABLES BY PRODUCT BREAK.ON PRODUCT TOTAL  
UNIT.PRICE
```

TRANSPORT

Use in a Retrieve sentence to display the last value for a specified field in a set of records. Its syntax is as follows:

```
TRANSPORT field.expression
```

field.expression specifies the name of a field or an EVAL expression for which you want to list the last value.

If the field uses the default column header in the report, the TRANSPORT keyword is added before the column header.

If you specify a multivalued field, Retrieve treats each value separately and lists the last value of the multivalues.

If you use the TRANSPORT keyword with a breakpoint keyword ([BREAK.ON](#) or [BREAK.SUP](#)), Retrieve displays the last value for each breakpoint in addition to the overall last value at the bottom of the report.

For example, to display all the last names in the PERSONNEL file, listing the last value at the bottom of the report, enter the following:

```
>LIST PERSONNEL TRANSPORT LAST.NAME
```

TRAP

Use with [RUN](#) and [RAID](#) to cause the RAID debugger to be entered when a warning message appears during run time.

TRUNCATE

Use with [DIVER.T.OUT](#) to delete any remaining part of an existing record which has not been overwritten by the current output diversion. The end of the record is at the end of the current output diversion. The TRUNCATE option should be specified when the output record already exists and the current diversion does not entirely overwrite the previous contents; otherwise the existing record is not cleared.

TTY.OFF

Use with [DIVER.T.OUT](#) to turn off terminal (tty) display of diverted output. User input is also turned off.

TTY.ON

Use with [DIVER.T.OUT](#) to turn on terminal output display if it has been disabled by the TTY.OFF keyword.

TYPE

Synonym for [FORM](#).

UNICODE

In NLS mode, use with [COPY](#), [CP](#), [CT](#), and [SET.FILE.MAP](#) to list records with each character represented by its 4-digit hexadecimal Unicode value. In the COPY command, use this option with the CRT or LPTR options.

–UNICODE

Use with [CP](#) and [CT](#) to list records with each character represented by its 4-digit hexadecimal Unicode value.

UNION

Use with **MERGE.LIST** to produce a select list whose elements are contained in *list1*, *list2*, or both.

UNIQUE

Use with **SAVING** to prevent duplicates in a list of values other than record ID values. UNIQUE gives an error message unless SAVING is also in the sentence. SAVING UNIQUE also sorts on the specified field.

To create a select list with only unique values, use a sentence like the following:

```
>SELECT PAYABLES WITH AMT.DUE > 1 SAVING UNIQUE  
NAMES
```

Use with the **BANNER** keyword in a **SETPTR (UNIX)** or **SETPTR (Windows Platforms)** sentence to specify a unique **&HOLD&** record at each invocation of SETPTR.

UNLIKE

Synonym for **NOT.MATCHING**.

UP

Use with **CHAP** to raise execution priority. If you are not in the UV account, CHAP UP has no effect.

UPDATING

Use with **COPY** to copy a record only if a record with the same record ID exists in the target file. The new record overwrites the old.

USER

Use with **LIST.READU** to list the active locks owned by a specific user.

Use with **PORT.STATUS** to limit the report to jobs owned by a specific user.

Use with **UNLOCK** to restrict lock removal to the locks of a particular user. You can get a user's number from the output of **LIST.READU**.

USERS

Use with **STATUS** to display information about all user processes.

USING

Use with Retrieve commands and SQL statements to specify an alternate dictionary to be used as the file dictionary. Its syntax is as follows:

USING [DICT | PDICT] *filename*

filename is the name of the file to use as the alternate dictionary. If you specify DICT or PDICT, you use the dictionary of *filename*, otherwise you use the data file. USING can be put anywhere in the Retrieve sentence but can only appear once.

Use with the **REVISE** command to specify a process definition to be used for prompting. The process must exist in the file **REVISE.PROCESSES**. To use the process ENTER.DICT, enter the following:

>REVISE DICT MYFILE USING ENTER.DICT

Use with the **RESIZE** command to specify a path specifying a work area for RESIZE to use when creating temporary files required to resize a file.

UTF8

Use with **ASSIGN,T.ATT**, **SET.FILE.MAP**, and **SET.SEQ.MAP** to specify the name of an NLS character set map. UTF8 is the same as NONE, but with system delimiters mapped to the Unicode Private Use Area.

VERIFIELD

Use in a ReVise sentence to display values from a verification file. VERIFIELD must be used with either the VERIFY or VERIFILE keywords. Its syntax is as follows:

VERIFIELD *display.field*

display.field is the name of the field in the verification file to display when the entered value matches a record ID in the verification file.

When you use VERIFIELD, you cannot use VERIFY or VERIFILE to verify that values do *not* exist in the verification file.

To display account number values from ACCOUNTS when entering data to PAYABLES, use a sentence like the following:

**>REVISE PAYABLES VENDOR VERIFY ACCOUNTS VERIFIELD
ACCT.NO**

VERIFILE

Synonym for VERIFY.

VERIFY

Use in a ReVise sentence to verify that the information you are entering for a field does or does not match a record ID of another file.

To enter only values that match record IDs in another file, use the following syntax:

REVISE *filename* [*fields*] *verifield* VERIFY *verifile*

filename is the file whose records you are adding or changing.

fields are the names of fields to prompt for.

verifield is either the name of a field in the current file whose data is being validated or the name of the record ID field (@ID) in the current file.

Parameter	Description
<i>field</i>	If you use a field name, ReVise verifies that the value entered in <i>verifield</i> is also a record ID in <i>verifile</i> .
<i>record.ID</i>	If you use the record ID field name, ReVise verifies that the entered value is <i>not</i> a record ID in <i>verifile</i> . If the ReVise sentence also includes a VERIFIELD expression, ReVise verifies that the record ID entered <i>is</i> a record ID in <i>verifile</i> and displays the value of the VERIFIELD display field.

verifield Parameters

verifile is the name of the verification file, whose record IDs validate the data entered in *verifield*.

A value entered at the VENDOR prompt must be a record ID in the ACCOUNTS file:

>REVISE PAYABLES VENDOR VERIFY ACCOUNTS

To enter in the INVENTORY file only values that are not record IDs in the PAYABLES file, use a sentence like the following:

>REVISE INVENTORY @ID VERIFY PAYABLES

In NLS mode, use with the [SET.FILE.MAP](#) command to check that the map is valid for the data in the file. If the new map name results in lost data, the map is not changed unless you also specify the FORCE option.

VERT

Synonym for VERTICALLY.

VERTICALLY

Use to display the output in a vertical format, with each field on a separate line. There is no spacing or formatting between lines. If a multivalued field is displayed, the information is displayed in a column. If multivalued fields are associated, the information is displayed next to each other in columns.

If the information is too wide to fit across the screen in columns, Retrieve automatically displays it in vertical format. For example:

**>LIST PAYABLES WITH PYMT.DATE EQ 6 "2/12/85"
VERTICALLY**

The VERTICALLY keyword suppresses the printing of breakpoint text.

WHEN

Use in a Retrieve sentence to limit output to values or subvalues in a multivalued field that meet specified criteria. Its syntax is as follows:

*WHEN myfield operator { values | field2 }
[[{ AND | OR } myfield operator] { values | field2 } ...]*

myfield is the multivalued field to be checked for the condition. *operator* is a relational operator.

values are one or more constants. If a value contains delimiters or spaces, you must enclose it in quotation marks. The constants are compared with the values in *myfield*. If you specify more than one value, the OR operator is assumed; that is, the condition:

myfield operator value1 value2

means:

myfield operator value1 OR myfield operator value2

If you use the NE or UNLIKE operator (or their synonyms), specify only one constant.

field2 is the name of a singlevalued or multivalued field whose values are compared to the values in *myfield*.

If you use more than one WHEN clause in a sentence, and they are not joined with AND or OR, AND is assumed; that is, the sentence:

LIST filename WHEN myfield operator value WHEN myfield operator value

means:

LIST filename WHEN myfield operator value AND WHEN myfield operator value

If you use WHEN and WITH in the same sentence, Retrieve first selects records according to the selection criteria specified in the WITH clause, then limits output to lines meeting the print-limiting criteria specified in the WHEN clause.

To list a subset of values in the multivalued INTERESTS field, use a sentence like the following:

>LIST SUN.MEMBER WHEN INTERESTS EQ FISHING

To list values with a balance due when the product is office supplies, use a sentence like the following:

**>LIST PAYABLES WITH BAL.DUE > 0 WHEN PRODUCTS EQ
OFFICE.SUP**

WITH

Use in a Retrieve sentence to select records whose data meets specified criteria. Its syntax is as follows:

WITH [**EVERY**] *field1 operator { values | field2 }*
[{ **AND** | **OR** } [**EVERY**] *field1 operator { values | field2 }* . . .]

field1 is the field to be checked for the condition. *operator* is a relational operator.

values are one or more constants. If a value contains delimiters or spaces, you must enclose it in quotation marks. The constants are compared with the values in *field1*. If you use the NE or UNLIKE operator (or their synonyms), specify only one constant.

field2 is the name of a field whose values are compared to the values in *field1*.

You can use more than one WITH expression in a sentence. If no parentheses are used, records are selected by their first WITH condition, then by the second, and so on. You can alter this order by using parentheses.

To select all records with balances over 50, use a sentence like the following:

>SELECT PAYABLES WITH BAL.DUE > 50

To list all records with balances over 50 and a list payment date before 2/1/95 or after 2/28/95, use a sentence like the following:

**>LIST PAYABLES WITH BAL.DUE > 50 AND (PYMT.DATE < 2/1/95
OR PYMT.DATE > 2/28/95)**

Selection expressions can be specified using a shorthand syntax that uses default settings. This makes the selection expression shorter but a little harder to read. The following table shows some examples of the shorthand syntax with longer equivalents. (In this syntax, *op* is any operator keyword, for example, GT or LT.)

Shorthand	Longer Equivalent
<i>op value</i>	WITH @ID <i>op value</i>
WITH <i>field</i>	WITH <i>field</i> NE ""
WITH <i>field1 field2</i>	WITH <i>field1</i> = <i>field2</i>
WITH NOT <i>field</i>	WITH <i>field</i> = ""
WITH <i>field op value</i> OR <i>op value</i>	WITH <i>field op value</i> OR <i>field op value</i>

Shorthand Syntax

If you do not specify a field, the search is performed for values in the @ID field.

>LIST SUN.MEMBER = 6100

This sentence is a terser version of LIST SUN.MEMBER WITH @ID = 6100.

If you do not specify an operator and a value, the criterion NE "" is supplied. It selects any record not having an empty string as the value in the field.

>LIST SUN.MEMBER WITH ZIP

This sentence is the same as LIST SUN.MEMBER WITH ZIP NE "".

If you specify a field and a value in the selection criteria, but no operator, “equals” is assumed. The following sentence selects records that have a value in the first field that is equal to the second field.

>LIST SUN.MEMBER WITH YR.JOIN YR.CANCEL

The keyword NOT negates the selection criteria. The following expression selects any record that has an empty string as the value in the specified field.

>LIST SUN.MEMBER WITH NOT ZIP

This sentence is the same as LIST SUN.MEMBER WITH ZIP EQ "".

Compound expressions have greater flexibility. For instance, in the following sentence you do not have to provide a field name in the second expression, as long as the field name is the same as in the first expression:

>LIST SUN.MEMBER WITH ZIP > 44000 AND < 66000

If the logical operator AND is omitted, OR is assumed.

WITHIN

Use with Retrieve commands to Retrieve one record with all its related subrecords. Its syntax is as follows:

WITHIN *filename* 'record.ID'

filename is the name of the file.

record.ID is the name of a record that has related subrecords. You can specify only one record ID in a query that uses the WITHIN keyword.

Subrecords provide a further description, or breakdown, of a record—for example, a bill-of-materials record. Subrecords are stored in the same file as the records they depend on.

To make a record a subrecord of another record, store the subrecord's ID as a value in a multivalued field containing subrecord IDs. You can add multiple subrecords to a record by storing their record IDs as multivalues in the subrecord field. And you can nest subrecords by storing their IDs in other subrecords.

Two conditions must be met for the WITHIN keyword to work. First, the record specified by the WITHIN keyword must contain a multivalued field whose values are record IDs (the subrecords). Second, field 6 of the file definition in the VOC file must contain the number of the subrecord field.

Each subrecord encountered is assigned a *level number*. The record ID specified in the WITHIN expression is Level 1. If this record has subrecords, they are assigned Level 2. If Level 2's subrecords have subrecords, they are assigned Level 3, etc., up to a maximum of 20 levels. Level numbers are listed in front of the record and its subrecords.

In this example, the multivalued field SUBREC contains subrecord IDs in the PRODUCT file. The Retrieve query lists record 801-77 and three levels of subrecords referenced by it.

```
>LIST WITHIN PRODUCT '801-77' PROD.NO DESC PRICE LOCATION SUBREC  
QOH ID.SUP
```

—XREF

Use with the **BASIC** command to generate a cross-reference table of statement labels and variable names used in the program.

User Records

Alphabetical Summary of User Records	3-3
INTR.KEY	3-4
QUIT.KEY	3-6
RELLEVEL	3-8
STACKWRITE	3-9
SUSP.KEY	3-10

This chapter describes every user record in the default IDEAL UniVerse flavor VOC file. An X in field 1 specifies a user record in the VOC file. User records are sometimes called X-descriptors. The user records are arranged alphabetically with each record starting on a new page. The user record name and a brief definition are at the top of the page, followed by a listing of the record and notes on use. The following sample shows a typical user record reference page.

USER RECORD

Brief description of the user record.

Listing

The format is as follows:

	USER.RECORD
001:	X
002:	<i>text</i>
<i>nnn</i> :	<i>text</i>

CAPITAL LETTERS indicate the text appears exactly as shown.

Italic indicate the kind of information in the field.

A vertical bar (|) between two elements indicates that either can be included.

Brackets ([]) enclose optional text.

Description

This section describes how to use the record and includes any applicable warnings.

Alphabetical Summary of User Records

The following table lists all user records described in this chapter.

Record	Description
INTR.KEY	Specifies the options available when the Intr (interrupt) key is pressed.
QUIT.KEY	Specifies the options available when the Quit key is pressed.
RELLEVEL	Specifies the current UniVerse revision level and account flavor.
STACKWRITE	Specifies whether the sentence stack is saved for the next login session.
SUSP.KEY	Specifies the options available when the Susp (suspend) key is pressed.

UniVerse User Records

INTR.KEY

INTR.KEY is an X record identifying the break options available when you press the Intr (interrupt) key.

Listing

```
                                INTR.KEY
001:      X
002:      options
```

options is a string of one or more of the characters A, C, D, L, and Q. If you specify only one option, the action associated with that option occurs when you press the Intr key.

Description

If you specify more than one option, a message like the following appears when you press the Intr key:

```
Break: Option (A,C,L,Q,?) =
```

The user can then select an option. Pressing ? at the prompt displays the actions of the available options.

Option	Description
A	Aborts the current procedure and returns to the system prompt.
C	Continues executing the current procedure. You can also continue the current procedure by pressing ENTER.
L	Aborts the current procedure and runs the login procedure as specified in the LOGIN paragraph.
Q	Aborts the current procedure and exits UniVerse entirely.
D	If the user is running a BASIC program, the D option aborts the program and enters the debugger.

BREAK Options



You can set and display the Intr key with the [PTERM \(UNIX\)](#) or the [PTERM \(Windows Platforms\)](#) command.

Note: *UniVerse loads the values for INTR.KEY when UniVerse starts. Any modifications to the values do not take effect until you restart UniVerse.*

QUIT.KEY

QUIT.KEY is an X record identifying the break options available when you press the Quit key.

Listing

```
                QUIT.KEY
001:           X
002:           options
```

options is a string of one or more of the characters A, C, L, and Q. If you specify only one option, the action associated with that option occurs when you press the Quit key.

Description

If you specify more than one option, a message like the following appears when you press the Quit key:

```
Break: Option (A,C,L,Q,?) =
```

You can then select an option. Pressing ? at the prompt displays the actions of the available options.

Option	Description
A	Aborts the current procedure and returns to the system prompt.
C	Continues executing the current procedure. You can also continue the current procedure by pressing ENTER.
L	Aborts the current procedure and runs the login procedure as specified in the LOGIN paragraph.
Q	Aborts the current procedure and exits UniVerse entirely.
D	If the user is running a BASIC program, the D option aborts the program and enters the debugger.

QUIT.KEY Options

You can set and display the Quit key with the [PTERM \(UNIX\)](#) or the [PTERM \(Windows Platforms\)](#) command.



Note: *UniVerse loads the values for QUITKEY when UniVerse starts. Any modifications to the values do not take effect until you restart UniVerse.*

RELLEVEL

RELLEVEL is an X record that identifies the revision level of the VOC file.

Listing

```
RELLEVEL
001:  X
002:  version.number
003:  flavor
```

version.number is the latest version number of the account.

flavor is the UniVerse flavor for which the current VOC file was configured. The BASIC run machine uses this field to determine the account flavor for run-time flavor dependencies.

Description

When you invoke UniVerse, an internal version number is compared to the version number in field 2 of the RELLEVEL. If they are different, UniVerse prints the following message:

Your VOC is version [*old*], update to [*new*]?

If you enter **Y**, the VOC is automatically updated to the latest version (see the [UPDATE.ACCOUNT](#) command).

STACKWRITE

STACKWRITE is an X record that indicates whether the sentence stack processor should maintain the sentence stack after you log out.

Listing

	STACKWRITE
001:	X
002:	ON OFF

If field 2 of this record contains ON, the sentence stack is saved on logout. If this field contains OFF, the sentence stack is not saved.



***Note:** UniVerse loads the values for STACKWRITE when UniVerse starts. Any modifications to the values do not take effect until you restart UniVerse.*

SUSP.KEY

SUSP.KEY is an X record identifying the break options available when the user presses the Susp (suspend) key.

Listing

```
                SUSP.KEY
001:           X
002:           options
```

options is a string of one or more of the characters A, C, L, and Q. If you specify only one option, the action associated with that option occurs when you press the Susp key.

Description

If you specify more than one option, a message like the following appears when you press the Susp key:

```
Break: Option (A,C,L,Q,?) =
```

You can then select an option. Pressing ? at the prompt displays the actions of the available options.

Option	Description
A	Aborts the current procedure and returns to the system prompt.
C	Continues executing the current procedure. You can also continue the current procedure by pressing ENTER.
L	Aborts the current procedure and runs the login procedure as specified in the LOGIN paragraph.
Q	Aborts the current procedure and exits UniVerse entirely.
D	If the user is running a BASIC program, the D option aborts the program and enters the debugger.

SUSP.KEY Options



You can set and display the Susp key with the [PTERM \(UNIX\)](#) or the [PTERM \(Windows Platforms\)](#) command.

Note: *UniVerse loads the values for SUSP.KEY when UniVerse starts. Any modifications to the values do not take effect until you restart UniVerse.*

Local and System Files

&COMO&	4-3
&DEVICE&	4-4
&ED&	4-6
&HOLD&	4-8
&MAP&	4-9
&PARTFILES&	4-10
&PH&	4-11
&SAVEDLISTS&	4-12
&TEMP&	4-13
&UFD&	4-14
&UNICODE.FILE&	4-15
APP.PROGS	4-16
BLTRS	4-17
DICT.DICT	4-18
DICT.PICK	4-19
NEWACC	4-20
REVISE.PROCESSES	4-21
STAT.FILE	4-22
SYS.HELP	4-23
SYS.MESSAGE	4-24
UNIVERSE.MENU.FILE	4-25
UNIVERSE.STAT.FILE	4-26
UNIVERSE.VOCLIB	4-27
USER.HELP	4-28

This chapter describes the local and system files used by UniVerse processes. Each UniVerse account has access to these files. Some of these files are local, that is, they reside in the user’s account. Others are system files located in the UV account. The VOC file for each UniVerse account contains pointers to the system files. System files are write protected so that only the UniVerse Administrator can change them.

Local Files	System Files
&COMO&	&DEVICE&
&ED&	&MAP&
&HOLD&	&PARTFILES&
&PH&	APP.PROGS
&SAVEDLISTS&	BLTRS
&TEMP&	DICT.DICT
&UFD&	DICT.PICK
&UNICODE.FILE&	NEWACC
STAT.FILE	REVISE.PROCESSES
USER.HELP	SYS.HELP
	SYS.MESSAGE
	UNIVERSE.MENU.FILE
	UNIVERSE.STAT.FILE
	UNIVERSE.VOCLIB
Local and System Files	

&COMO&

Description

&COMO& is a local type 1 file that UniVerse uses to hold command output (COMO) records. You can use the COMO command to save command output in a record in the &COMO& file as it is being displayed on your terminal. The COMO command creates an &COMO& file if one does not exist.

As with any other type 1 file, you can examine the individual records and change them using the Editor. To edit a COMO record, use the following syntax:

ED &COMO& *record*

record is the name you gave the COMO record when you used the COMO command.

To print a COMO record using the SPOOL option with the COMO command, use the following syntax:

COMO SPOOL *record*

Or use the SPOOL command with the following syntax:

SPOOL &COMO& *record*

For more information, see the [COMO](#) command.

&DEVICE&

Description

&DEVICE& is a hashed system file that is used by the [ASSIGN](#) command to define the device to be assigned. Each record comprises a device name, a description of that device, and the system filename for the device. You cannot assign a device unless it has an entry in the &DEVICE& file.

The file dictionary of &DEVICE& is shown here to help you interpret the fields of the &DEVICE& file:

DICT &DEVICE& 04:03:04PM 15 Jul 1997 Page 1

		Type &			
Field.....	Field.	Field.....	Conversion..	Column.....	Output
Depth &					
Name.....	Number	Definition...	Code.....	Heading.....	Format
Assoc..					
@ID	D	0		DEVICE	10L S
DESC	D	1		Description	20T S
FILE	D	2		UNIX pathname	25L S
BLOCK	D	3		Block size	5R S
TYPE	D	4		Device type	2L S
LOCK	D	5		Lock files	25L M
REWDEV	D	6		Rewind device	25L S
NREWDEV	D	7		NoRewind device	25L S
BACKUP	D	8		cpio-backup	50L S
RESTORE	D	9		cpio-restore	50L S
SKIP	D	10		Skip file	50L S
REWIND	D	11		Rewind tape	50L S
OFFLINE	D	12		Tape offline	50L S
WRITEAT	D	13		Field 13	1L S
CLOSE	D	14		Field 14	1L S
EOFREAD	D	15		Field 15	1L S
IBS	D	16		Input Blksize	6R S
ROTFLG	D	17		Rotate	1L S
OTHER	D	18		Flag	
NLSMAP	D	19		Field 18	70T M
				NLS	32L S
				map name	
NDELAY	D	20		N_DELAY	1L S
				Flag	
OPTIONS	I	SUBR (		sp.config optio	30T S

```

                                "*DISP.SP.OPT          ns
                                S", @RECORD )
@                               PH  DESC FILE
                                BLOCK TYPE
                                LOCK REWDEV
NREWDEV
                                BACKUP
                                RESTORE SKIP
                                REWIND
                                OFFLINE
                                WRITEAT CLOSE
                                EOFREAD IBS
                                ROTFLG NLSMAP
RESERVED          X             Fields 6 - 16
                                reserved for
                                SP.CONFIG.MNT

24 records listed.

```

In NLS mode, you can associate a map name with a printer or any other device defined in the &DEVICE& file. To do this, add the map name in field 19 of the device's record in &DEVICE&.

- If a device has a specific map defined in &DEVICE&, all input and output for the device uses the map.
- If a device does not have a specific map defined in &DEVICE&, it uses the default specified in the *uvconfig* file. The defaults are specified in the following parameters:
 - NLSDEFPTRMAP, for printers
 - NLSDEFTERMMAP, for terminals
 - NLSDEFDEVMAP, for other devices

The map name can be overridden by certain commands, such as [ASSIGN](#). For more information, see the *UniVerse NLS Guide*.

&ED&

Description

&ED& is a local type 1 file that stores command sequences that can be used by the Editor during later editing sessions. Several of the commands in the Editor let you save, list, execute, recall, and delete Editor command records in the &ED& file. You must use the [CREATE.FILE](#) command to create an &ED& file before storing records in it.

You can create records in the &ED& file by using the Editor .S (Save) command to save commands from the Editor command stack. The following Editor commands manipulate the &ED& file:

Command	Description
.D <i>name</i>	Deletes <i>name</i> from &ED&.
.L <i>name</i>	Lists <i>name</i> from &ED&.
.R <i>name</i>	Loads <i>name</i> from &ED& into the current Editor command stack.
.S <i>name s e</i>	Saves lines <i>s</i> through <i>e</i> from the command stack as <i>name</i> in &ED&.
.S <i>n name</i>	Saves lines 1 through <i>n</i> from the command stack as <i>name</i> in &ED&.
.X <i>name</i>	Executes the saved command record <i>name</i> from &ED&.

Editor Commands

name is the record ID of the stored command record in the &ED& file. Here is an example of a record MARK.BLOCK in &ED&:

```
0001: E
0002: <
0003: P 3
0004: >
```

You could execute this command by entering **.X MARK.BLOCK** at the Editor prompt. It would define a three-line block, beginning at the current line.

You can also create and modify records in the &ED& file using the Editor as you would for any other file. If you create a record in the &ED& file using the Editor, you must enter the letter E on line 1. This indicates that the record is an executable Editor command. The rest of the record must consist of valid Editor commands.

To avoid confusion, store only Editor command records in &ED&.

&HOLD&

Description

&HOLD& is a local type 1 file used to hold spooled reports. If &HOLD& does not exist in the account, [SETPTR \(UNIX\)](#) or [SETPTR \(Windows Platforms\)](#) creates it when you use the SETPTR command to set a logical print channel to mode 3. Spooled reports to that channel become records in the &HOLD& file.

Once a report has been sent to &HOLD&, you can edit, delete, or spool the report by using the Editor. The report remains in &HOLD& until you delete it or until another report with the same name is written to &HOLD&.

The dictionary of &HOLD& contains a NEXT.HOLD record, which is an X-type item used by the spooler to create unique print file names. Field 2 of NEXT.HOLD contains a sequence number from 1 through 9999.

In NLS mode, the spool queue directory holds data in UniVerse internal format. When data reaches the printer, it is mapped to an external character set using the appropriate map for the device. However, the default map associated with this file is NONE, unless the directory existed before you installed NLS. When you spool to a hold file, the spooler stores the data using the map associated with the &HOLD& directory. The data is then mapped again when it reaches the printer. For more information, see the *UniVerse NLS Guide*.

&MAP&

Description

The [MAKE.MAP.FILE](#) command builds a hashed UniVerse file called &MAP&. &MAP& contains records that describe the contents of the system catalog space and can be used by Retrieve for search and retrieval purposes. You can produce reports about the programs in the catalog space, so that you can manage the space better.

Usually the &MAP& file is created in the UV account with remote pointers from other accounts. To create a local &MAP& file in your own account, follow these steps:

1. Delete or change the name of the &MAP& entry in your VOC file.
2. Create &MAP& in your account using [CREATE.FILE](#) to create the data file.
3. Change the &MAP& VOC entry to have field 3 point to the file dictionary (D_&MAP&) in the UV account. The &MAP& VOC entry should look something like this:

0001: F
0002: &MAP&
0003: /usr/ibm/uv/D_&MAP&
4. Use MAKE.MAP.FILE.

The &MAP& file becomes out-of-date as soon as the catalog space is changed by a [CATALOG](#) or [DELETE.CATALOG](#) command. If you are using &MAP& to assist in managing the catalog space, use MAKE.MAP.FILE immediately before creating a report. By creating a new &MAP& file, you ensure that the information describing the catalog space is current.

&PARTFILES&

Description

&PARTFILES& is a hashed system file used to store records defining a part file that belongs to a distributed file. Each time you use the `DEFINE.DF` command to define a distributed file, the record for each part file is created in the &PARTFILES& file. The &PARTFILES& file resides in the UV account.

Each part file record in the &PARTFILES& file contains the following:

- The path of the file
- A list of VOC records referencing the part file path
- A list of corresponding UniVerse account paths in which the VOC records reside
- The part number of the part file
- The partitioning algorithm

For more information, see the [DEFINE.DF](#) command.

&PH&

Description

&PH& is a local type 1 file used to store the output produced by a task started with the **PHANTOM** command. The PHANTOM command starts a background process. PHANTOM automatically creates a &PH& file record every time you start a phantom process. All terminal output produced by the PHANTOM process is stored in the record. You can verify the operation of a phantom process after the process has completed by examining the &PH& file record. You can also edit or spool the &PH& file records.

When you no longer need an &PH& file record, delete it from the &PH& file.

Each record in the &PH& file has a record ID consisting of the following:

verb_time_date

verb is a phantom verb or command. It is taken from a PHANTOM sentence and is the first word of the process to be executed as a phantom task. For example, the record ID is SORT_*time_date* for the following sentence:

PHANTOM SORT VENDOR

time is generated by the UniVerse BASIC INT function (TIME()) which produces an integer indicating the time that the process started.

date is generated by the BASIC DATE function(), the internal date. This suffix guarantees the uniqueness of &PH& file records in an account, even if several users begin the same phantom process at nearly the same time. To ensure unique record IDs, use a **SLEEP** command between PHANTOM commands.

If &PH& does not exist in the account when you use PHANTOM, the command creates the &PH& file. &PH& is a type 1 file and can be accessed from UniVerse or from your operating system.

&SAVEDLISTS&

Description

&SAVEDLISTS& is a local type 1 file used to store the lists referenced by the [SAVE.LIST](#), [EDIT.LIST](#), [GET.LIST](#), [COPY.LIST](#), [MERGE.LIST](#) and [DELETE.LIST](#) commands. &SAVEDLISTS& also stores the current sentence stack when you exit UniVerse, and stores the lists created by the VVOC command. Each list is a record in the file.

If you save a list using the [SAVE.LIST](#) command, the list name is the name that you specified, or, if you do not specify a name, the list name is &TEMP`port#`&, where `port#` is the terminal number, as displayed by the WHO command.

The VVOC command potentially creates three select lists: &VOC.ONLY, &NEWACC.ONLY, and &VOC.DIFF. These lists are stored in &SAVEDLISTS& and can be retrieved using the [GET.LIST](#) command and any other commands that are used with select lists. For a discussion of select lists, see the *Guide to Retrieve*.

When you log out and field 2 of the STACKWRITE entry in your VOC file is ON, the current sentence stack is saved in a list named &&S.`username.port#`&. The `username` is your user name, and `port #` is the terminal number. When you log in again from the same port and the same account, this becomes the current sentence stack.

You can use the [SAVE.STACK](#) command to save the current sentence stack to the &SAVEDLISTS& file under any name you like. Simply enter the SAVE.STACK command followed by the name you want to save the stack under. Use the GET.STACK command to retrieve any saved sentence stack and load it as the current sentence stack. The previously loaded stack is replaced by the retrieved stack. For more information about sentence stacks, see *UniVerse System Description*.

You can use the UniVerse Editor to examine and change records in &SAVEDLISTS&. If &SAVEDLISTS& does not exist when you use SAVE.LIST, the command automatically creates the type 1 file.

&TEMP&

Description

&TEMP& is a local type 1 file used by the VOC update process to store VOC items that are not defined correctly or that conflict with new items of the same name. Each of these items is stored as a record in the &TEMP& file when you use [CLEAN.ACCOUNT](#) or when you update your account to a new release. You can examine these items in &TEMP& and decide to keep them or delete them.

If the update process finds an item in your VOC file that has the same name as a new item in the release but has different contents, the process moves the item that already exists in your VOC file to &TEMP& and replaces it with the new item.

You can use the UniVerse Editor to examine the records in &TEMP&.

&UFD&

Description

&UFD& is a type 1 file that is the directory where your UniVerse account is located. Although it is a local file in that it references the local directory, its file dictionary (D_&UFD&) is a system file, located in the UV account. The &UFD& file entry in the VOC allows various UniVerse processes to access files in your current directory. The files in the current directory are accessed as records of the &UFD& UniVerse file. In many cases, normal user-level UniVerse processes such as the Editor, ReVise, or Retrieve commands cannot produce meaningful results by accessing &UFD& records. You can use the &UFD& file to access ASCII files by specifying the filename as a record in &UFD&. For example:

ED &UFD& mbox

[CLEAN.ACCOUNT](#) is an example of a command that uses the &UFD&. CLEAN.ACCOUNT uses &UFD& to search for file names that are not defined in the VOC file for that account.

&UNICODE.FILE&

Description

&UNICODE.FILE& is a local dynamic file created in NLS mode, with a map name of NONE.

The [UNICODE.FILE](#) command uses &UNICODE.FILE& under certain circumstances when it converts a mapped file to an unmapped file (using the internal character set). If any characters cannot be converted using the resulting output map, the records are instead written to the &UNICODE.FILE& file.

The conversion process reports the number of records written to the &UNICODE.FILE&. Their record IDs are recorded in a saved select list named BAD.IDS.

If you specify the TEST qualifier, no records are written to the &UNICODE.FILE&, but the IDs of any records that the process is unable to convert are saved in the BAD.IDS select list.

APP.PROGS

Description

APP.PROGS is a type 1 system file used to store the source of programs that IBM neither supports nor warrants. The object code for these programs is stored in the corresponding type 1 file, APP.PROGS.O.

BLTRS

Description

BLTRS is a hashed system file that contains the output for the [BLOCK.TERM](#) and [BLOCK.PRINT](#) commands.

The first field in the BLTRS file is the character width. The second through *n*th records are multivalued fields of the form [B or C]1v2v3v4v5, and so on. B means start with blanks, C means start with the character value you are printing (for example, the letter W is printed using Ws). Each value specifies how many of the characters to print. You alternate printing (B/C) with each multivalued field. The multivalued fields should add up to the character width.

DICT.DICT

Description

DICT.DICT is a system file dictionary that functions as the master file dictionary for Retrieve processing of all the other UniVerse dictionaries.

DICT.PICK

Description

DICT.PICK is a system file dictionary that functions as a master file dictionary for Retrieve processing of all the Pick-style dictionaries.

NEWACC

Description

NEWACC is a hashed system file used to build or update user VOC files. NEWACC comprises multiple data files, one for each flavor account. Each data file is a template for the VOC file of a particular flavor of a UniVerse account, used to create the initial VOC file in a new UniVerse account.

Once a new VOC file has been built in your account from NEWACC, you can add or delete the items that are necessary for your application.

Your UniVerse Administrator can add or delete commands from the NEWACC files to customize them for your particular installation.

When a new release of UniVerse is installed, all existing accounts must be updated. When you enter an account after a new release has been installed, a message is displayed on your terminal asking whether you want to update your account. If you enter yes, the update takes place, and the new information from NEWACC is installed in your VOC file.

New features could duplicate names of entries that you have created in your VOC file. The update process saves these duplicate entries in the &TEMP& file. You can examine the entries in the &TEMP& file and decide about keeping them. (A discussion about the [&TEMP&](#) file appears earlier in this chapter.)

REVISE.PROCESSES

Description

REVISE.PROCESSES is a hashed system file that contains the templates used by ReVise in building dictionaries, menus, and selectors for menus.

STAT.FILE

Description

STAT.FILE is a local hashed file that is used to store the results of the command [ACCOUNT.FILE.STATS](#).

The ACCOUNT.FILE.STATS command with the LOCAL option gathers and stores statistics in the STAT.FILE file.

The [LIST.FILE.STATS](#) command with the LOCAL option generates a report of the file statistics stored in the STAT.FILE. This lets you check the efficiency of the selected files all at once instead of issuing an individual command on each file.

SYS.HELP

Description

SYS.HELP is a hashed system file. It contains help text for commands, functions, keywords, and so on. SYS.HELP comprises the following data files:

- BASIC.HELP
- BCI.HELP
- CONV.HELP
- PICK.HELP
- SYS.HELP
- SQL.HELP

Each record in the appropriate SYS.HELP file contains descriptive text for a standard VOC file entry. When you use the HELP command for an item, the SYS.HELP record for that item is displayed. For more information, see the [HELP](#) command.

You can also use Retrieve commands on the SYS.HELP files to provide written documentation of the VOC file entries.

You can create a local help file in your own account. Call the file USER.HELP. See [USER.HELP](#) later in this chapter.

SYS.MESSAGE

Description

SYS.MESSAGE is a hashed system file. The records in the SYS.MESSAGE file contain the UniVerse system messages.

UNIVERSE.MENU.FILE

Description

UNIVERSE.MENU.FILE is a hashed system file. It is the default menu file and contains the menu for creating new menus. You should use the dictionary of this file as the file dictionary for any local menu files you create. To do this, create only the data file and then use the Editor to copy field 3 from the UNIVERSE.MENU.FILE entry:

```
>CREATE.FILE DATA MENU.FILE 3 1 2
Creating file "MENU.FILE" as Type 3, Modulo 1, Separation 2.
>ED VOC MENU.FILE
3 lines long.
----: 3
0003:
Bottom at line 3.
----: D
Bottom at line 2.
----: LOAD UNIVERSE.MENU.FILE
Starting line/field number - 3
Ending   line/field number - 3
0003: /usr/ardent/uv/D_MENU.FILE
Bottom at line 3.
----: FI
```

UNIVERSE.STAT.FILE

Description

UNIVERSE.STAT.FILE is a hashed system file in the UV account that is used to store the results of an [ACCOUNT.FILE.STATS](#) command.

The ACCOUNT.FILE.STATS command gathers and stores statistics in the UNIVERSE.STAT.FILE file.

The [LIST.FILE.STATS](#) command with no options generates a default report of the file statistics stored in the UNIVERSE.STAT.FILE in the UV account.

For more information, see [STAT.FILE](#) earlier in this chapter.

UNIVERSE.VOCLIB

Description

UNIVERSE.VOCLIB is a hashed system file. It is used to store long commands delivered as standard UniVerse commands that would otherwise appear in the VOC file, to keep the size of the VOC file down and the size of VOC entries relatively uniform. Entries in the VOC file point to paragraphs or procs in the UNIVERSE.VOCLIB file.

USER.HELP

Description

USER.HELP is a local help file, similar in use and format to SYS.HELP. USER.HELP can be used to document the VOC file entries that you create. USER.HELP is not supplied with UniVerse, but you can create by using [CREATE.FILE](#).

To create USER.HELP for your account, use the dictionary of the SYS.HELP file and follow the format of the records in the SYS.HELP file. You can either copy the records from the SYS.HELP dictionary to the USER.HELP dictionary, or set up the USER.HELP VOC entry so that it points to the SYS.HELP dictionary in the UV account. Otherwise, the HELP command will not work correctly.

The [HELP](#) command first displays the SYS.HELP record. HELP then searches for an item in the USER.HELP file if a USER.HELP file is in your account. If a record is found, it is displayed. USER.HELP can be used to supplement the information in SYS.HELP since both descriptions are displayed when you use HELP.

When creating records in USER.HELP, be sure to use the same format as that used in SYS.HELP.

UniVerse Commands

Quick Reference

This appendix briefly describes the UniVerse commands described in detail in Chapter 1, “[UniVerse Commands](#).” The commands are grouped according to use as follows:

- Sentence stack commands
- Redirecting output
- Paragraph commands
- Retrieve commands
- Creating and manipulating select lists
- Managing UniVerse files
- Managing records
- Managing UniVerse file structures
- Managing secondary indexes
- Managing BASIC programs
- Arithmetic commands
- Account commands
- VOC file commands
- Managing menus
- Printer commands
- Terminal commands
- Tape commands
- Managing the system
- Managing locks
- Managing transaction logging

- NLS commands

Command descriptions can also be found online in the VOC file.

Sentence Stack Commands

Command	Description
.A	Appends text to a sentence.
.C	Changes text in a sentence.
.D	Deletes a sentence from the sentence stack.
.I	Inserts a sentence in the sentence stack.
.L	Lists the sentences in the sentence stack, or lists the contents of a VOC file entry.
.R	Recalls a sentence to the first line of the sentence stack.
.S	Saves sentences in a VOC file entry.
.U	Changes a sentence to uppercase.
.X	Executes a sentence from the sentence stack.
.?	(Question mark) Terminates a sentence without executing it.
GET.STACK	Retrieves a sentence stack from the &SAVEDLISTS& file and puts it in the current sentence stack.
SAVE.STACK	Saves the current sentence stack in the &SAVEDLISTS& file.

Sentence Stack Commands

Redirecting Output

Command	Description
COMO	Manipulates COMO (command output) records.
DIVERT.OUT	Diverts the output of subsequent commands to a record in a type 1 file.
HUSH	Suppresses terminal messages during processing.
REFORMAT	Redirects Retrieve output to a file or to tape.
SREFORMAT	Redirects Retrieve output to a file or to tape, with records sorted by record ID.
T.DUMP	Copies files from disk to tape.
T.LOAD	Copies files from tape to disk.

Redirecting Output Commands

Paragraph Commands

Command	Description
<code>.S</code>	A sentence stack command that saves sentences in a VOC file entry.
<code>*</code>	(Asterisk) Indicates that the text that follows is part of a comment.
<code><<...>></code>	Displays a prompt and requests an input value when a sentence is executed.
<code>CLEARDATA</code>	Clears a data stack.
<code>CLEARPROMPTS</code>	Sets the value of in-line prompts to empty strings.
<code>DATA</code>	Specifies a response to input requests.
<code>DISPLAY</code>	Prints a line on the terminal.
<code>GO</code>	Transfers execution to a subsequent sentence in a paragraph.
<code>IF</code>	Conditionally executes statements in a paragraph.
<code>LOOP</code>	Defines the beginning of a range of sentences to be repeatedly executed.
<code>REPEAT</code>	Ends a LOOP...REPEAT loop in a paragraph.

Paragraph Commands

Retrieve Commands

Command	Description
CHECK.SUM	Gets statistical information on values in a particular field for one or more records in a file.
COUNT	Counts the records in a file.
ESEARCH	Creates a select list of records that contain an occurrence of a specified string.
LIST	Searches for and displays data from records in a file.
LIST.ITEM	Displays full listings of selected records.
LIST.LABEL	Displays records in a format suitable for mailing labels and other block listings.
REFORMAT	Redirects Retrieve output to a file or to tape.
SEARCH	Creates a select list of records that contain an occurrence of a specified string.
SELECT	Creates a list of records that meet specified selection criteria.
SORT	Lists selected records sorted by record ID.
SORT.ITEM	Displays full listings of selected records in sorted order.
SORT.LABEL	Displays items in a format suitable for mailing labels and other block listings.
SREFORMAT	Redirects Retrieve output to a file or to tape, with records sorted by record ID.
SSELECT	Creates a sorted list of records that meet specified selection criteria.
STAT	Displays numeric statistics for fields in a file.
SUM	Adds numeric values in fields of records that meet specified selection criteria.
T.DUMP	Copies files from disk to tape.
T.LOAD	Copies files from tape to disk.

Retrieve Commands

Creating and Manipulating Select Lists

Command	Description
CLEARSELECT	Clears an active select list.
COPY.LIST	Copies a saved list from the &SAVEDLISTS& file to another file.
DELETE.LIST	Removes a select list from the &SAVEDLISTS& file.
EDIT.LIST	Edits the contents of a saved list.
ESEARCH	Creates a select list of records that contain an occurrence of a specified string.
FORM.LIST	Creates a select list from a record.
GET.LIST	Activates a saved select list.
LIST.DIFF	Compares two saved lists and puts the elements from list one that are not found in list two in a third list.
LIST.INTER	Compares two saved lists and puts the elements from list one that are also found in list two in a third list, without redundancy.
LIST.UNION	Compares two saved lists and puts in a third list the elements from list one followed by the elements from list two that are not found in list one, without redundancy.
MERGE.LIST	Merges two numbered select lists using relational set operations and puts the result in a third select list.
NSELECT	Creates a select list by removing elements from a previously created select list.
QSELECT	Create a select list containing fields of one or more records.
SAVE.LIST	Saves an active select list.

Select List Commands

Command	Description
SEARCH	Creates a select list of records that contain an occurrence of a specified string.
SELECT	Creates a list of records that meet specified selection criteria.
SSELECT	Creates a sorted list of records that meet specified selection criteria.
Select List Commands (Continued)	

Managing UniVerse Files

Command	Description
CD	Compiles I-descriptors in a dictionary.
CLEAR.FILE	Removes all records in a file.
CNAME	Changes the name of a file or record in any UniVerse file.
COMPILE.DICT	Compiles I-descriptors in a dictionary.
COMPILE.DICTS	Compiles I-descriptors in all dictionaries in an account.
CONVERT.SQL	Converts a UniVerse file to an SQL table.
CREATE.FILE	Creates a UniVerse file.
DEFINE.DF	Defines or modifies a distributed file.
DELETE.FILE	Removes a data file or a file dictionary.
DLIST	Lists I-descriptor object code.
LIST.SICA	Displays SICA information about an SQL table.
REFORMAT	Redirects Retrieve output to a file or to tape.
SET.FILE	Creates a Q-pointer in the VOC file to a remote file.
SETFILE	Creates a file pointer.
SREFORMAT	Redirects Retrieve output to a file or to tape, with records sorted by record ID.
VERIFY.SQL	Verifies and fixes SQL catalog inconsistencies.

UniVerse File Commands

Managing Records

Command	Description
CLEAR.FILE	Removes all records in a file.
CNAME	Changes the name of a file or record in any UniVerse file.
CONVERT.VOC	Converts Pick or Prime dictionary records to a format compatible with UniVerse.
COPY	Copies records to a file, a printer, or the terminal.
COUNT	Counts the records in a file.
CP	Prints a record on the system printer.
CT	Displays a record on the terminal.
DELETE	Removes records from a file.
ED	Invokes the UniVerse Editor.
EXCHANGE	Switches the content of two records.
LIST.ITEM	Displays full listings of selected records.
RECORD	Runs a trial hashing of a record in a file.
REVISE	Adds, changes, and deletes file records.
SORT.ITEM	Displays full listings of selected records in sorted order.
SPOOL	Spools records to the printer.
UV.VI	Invokes the UNIX <i>vi</i> editor using the file naming syntax. This command is not supported on Windows platforms.
VI	Invokes the UNIX <i>vi</i> editor. This command is not supported on Windows platforms.

Record Commands

Managing UniVerse File Structures

Command	Description
ACCOUNT.FILE.STATS	Gathers file statistics.
ANALYZE.FILE	Gets information about a dynamic file.
CONFIGURE.FILE	Changes the parameters of an existing dynamic file.
CONVERT.SQL	Converts a UniVerse file to an SQL table.
FILE.STAT	Gets statistical information about a file.
FILE.USAGE	Displays a report of file usage statistics for any hashed file.
FILE.USAGE.CLEAR	Resets the statistics displayed by the FILE.USAGE command for a file.
FORMAT.CONV	Changes storage format of UniVerse files or object code.
GROUP.STAT	Produces a summary of the record distribution for a file.
GROUP.STAT.DETAIL	Gets more information than GROUP.STAT provides.
HASH.AID	Tries one or more file type, modulo, and separation combinations for even record distribution.
HASH.HELP	Determines the most efficient file structure for a file.
HASH.HELP.DETAIL	Gets more information than HASH.HELP provides.
HASH.TEST	Experiments with the file type, modulo, and separation recommended by HASH.HELP or HASH.HELP.DETAIL and produces a summary of the new record distribution.

File Structure Commands

Command	Description
HASH.TEST.DETAIL	Experiments with a file type and modulo recommended by HASH.HELP or HASH.HELP.DETAIL and produces a detailed summary of the new record distribution, including the size of each group.
LIST.DF	Lists paths and part numbers of part files belonging to a distributed file.
LIST.FILE.STATS	Displays file statistics gathered by ACCOUNT.FILE.STATS .
LIST.SICA	Displays SICA information about an SQL table.
REBUILD.DF	Verifies and fixes distributed file inconsistencies.
RECORD	Runs a trial hashing of a record in a file.
RESIZE	Reorganizes a file.
uvfixfile	Verifies and fixes broken files.
UVFIXFILE	Verifies and fixes broken files.
VERIFY.DF	Verifies the paths and part numbers of part files belonging to a distributed file.
VERIFY.SQL	Verifies and fixes SQL catalog inconsistencies.

File Structure Commands (Continued)

Managing Secondary Indexes

Command	Description
BUILD.INDEX	Builds a secondary index for a specified file.
CREATE.INDEX	Creates a secondary index for a file.
DELETE.INDEX	Deletes a secondary index from a file.
DISABLE.INDEX	Disables automatic updating of secondary key indexes of a file.
ENABLE.INDEX	Reenables the automatic updating of the secondary key indexes of a file.
LIST.INDEX	Displays information about a file's secondary key indexes.
SET.INDEX	Changes the characteristics of all the secondary indexes in a file.
UPDATE.INDEX	Updates the secondary key indexes of a file.

Index Commands

Managing BASIC Programs

Command	Description
BASIC	Compiles a BASIC program.
CATALOG	Catalogs a compiled BASIC program.
CLEARCOMMON	Resets the variables in the common area to zero.
CLEARDATA	Clears a data stack.
DECATALOG	Deletes a locally cataloged program from the VOC file.
DELETE.CATALOG	Deletes programs from the system or local catalog.
FANCY.FORMAT	Formats a BASIC source program.
FORMAT	Formats a BASIC source program.
IAM	Sets the effective account name.
RAID	Debugs a BASIC program.
RUN	Executes a compiled BASIC program.
VCATALOG	Compares BASIC object codes.
VLIST	Lists BASIC object code.
BASIC Program Commands	

Arithmetic Commands

Command	Description
PRIME	Finds the nearest prime numbers.
RADIX	Converts the number base.

Arithmetic Commands

Account Commands

Command	Description
ABORT	Aborts the current process.
ABORT.LOGIN	Aborts the current process and runs the LOGIN paragraph.
ACCOUNT.FILE.STATS	Gathers file statistics.
AUTOLOGOUT	Sets UniVerse to log you out automatically if you have not pressed a key within the specified time.
CHDIR	Switches to another UniVerse directory.
CLEAN.ACCOUNT	Performs account maintenance.
COMPILE.DICTS	Compiles all the dictionaries in an account.
CONVERT.ACCOUNT	Converts a Pick or Prime account from its original format to a format compatible with UniVerse.
CONVERT.SQL	Converts a UniVerse file to an SQL table.
CONVERT.VOC	Converts Pick or Prime dictionary records to a format compatible with UniVerse.
IAM	Sets the effective account name.
INITIALIZE.DEMO	Initializes the demonstration files CUSTOMERS, INVENTORY, and ORDERS.
LIST.FILE.STATS	Displays file statistics gathered by ACCOUNT.FILE.STATS .
LO	Exits UniVerse.
LOGIN	Initializes UniVerse account environment.
LOGOUT	Exits UniVerse.
LOGTO	Moves to another UniVerse account.
LOGTO.ABORT	Moves to another UniVerse account, and aborts the current process.

Account Commands

Command	Description
OFF	Exits UniVerse.
PASSWD	Sets or changes the password for the current account.
QUIT	Leaves UniVerse and returns to where UniVerse was invoked.
UPDATE.ACCOUNT	Updates the contents of the VOC file by replacing records in it with records from the system's NEWACC file.
uvbackup (UNIX) uvbackup (Windows Platforms)	Begins a daily, weekly, or full save of an entire system, or selected directories or files.
uvrestore (UNIX) uvrestore (Windows Platforms)	Restores an entire system, or selected directories, files, or records in UniVerse hashed files saved by <i>uvbackup</i> .
WHO	Displays the port number and account name.

Account Commands (Continued)

VOC File Commands

Command	Description
.L	A sentence stack command that lists the sentences in the sentence stack or the contents of a VOC file entry.
..S	A sentence stack command that saves sentences in a VOC file entry.
CLEAN.ACCOUNT	Performs account maintenance.
CONVERT.ACCOUNT	Converts a Pick or Prime account from its original format to a format compatible with UniVerse.
CONVERT.VOC	Converts Pick or Prime dictionary records to a format compatible with UniVerse.
LISTDOS	Lists DOS commands in the VOC file.
LISTF	Lists files defined in the VOC file.
LISTFL	Lists files stored in this account.
LISTFR	Lists files stored in remote accounts.
LISTK	Lists keywords stored in the VOC file.
LISTM	Lists menu selector records in the VOC file.
LISTO	Lists “other”-type keywords in the VOC file.
LISTPA	Lists paragraphs stored in the VOC file.
LISTPH	Lists phrases stored in the VOC file.
LISTPQ	Lists proc commands in the VOC file.
LISTR	Lists remote commands in the VOC file.
LISTS	Lists stored sentences in the VOC file.
LISTSL	Lists verbs stored in the VOC file that use a select list.
LISTUN	Lists UNIX commands in the VOC file.

VOC File Commands

Command	Description
LISTV	Lists verbs stored in the VOC file.
SET.FILE	Creates a Q-pointer in the VOC file to a remote file.
SETFILE	Creates a file pointer.
UPDATE.ACCOUNT	Updates the contents of the VOC file by replacing records in it with records from the system's NEWACC file.
VVOC	Verifies the contents of the VOC file by comparing it to the system's NEWACC file.
VOC File Commands (Continued)	

Managing Menus

Command	Description
MENU.DOC	Produces a menu listing that shows selection text, explanations, and the actual sentence, paragraph, or menu that is executed for each selection.
MENU.PIX	Prints an image of the menu on the printer.
MENUS	Creates, updates, and maintains menus.
MOTIF	Displays a UniVerse menu in Motif format.

Menu Commands

Printer Commands

Command	Description
ASSIGN	Assigns a device for exclusive use.
BLOCK.PRINT	Prints block characters on the printer.
CP	Prints a record on the system printer.
P.ATT	Requests control of a printer.
P.DET	Releases the printer from control.
PRINT.ADMIN	Invokes the print spooler queue administration menu. This command is not supported on Windows platforms.
SETPTR (UNIX) SETPTR (Windows Platforms)	Sets line printer spooler options.
SETPTR.DEFAULT	Changes the default values used by the SETPTR command over the entire system.
SP.ASSIGN (UNIX) SP.ASSIGN (Windows Platforms)	Sets the line printer spooler options for each of the 256 logical print channels.
SP.EDIT (UNIX) SP.EDIT (Windows Platforms)	Selects held spool files from the spool queue and sends them to the terminal or the printer.
SP.TAPE	Prints a report stored on a spooled tape.
SPOOL SPOOL (Windows Platforms)	Spools records to the printer.
TERM	Sets printer and terminal characteristics.
UNASSIGN	Releases control of a device.
<i>usa</i>	Administers the UniVerse spooler from a UNIX shell. This command is not supported on Windows platforms.

Printer Commands

Command	Description
<i>usd</i>	Starts the UniVerse spooler from a UNIX shell. This command is not supported on Windows platforms.
<i>usm</i>	Changes print job characteristics from a UNIX shell. This command is not supported on Windows platforms.
<i>usp</i>	Sends print jobs to the UniVerse spooler from a UNIX shell. This command is not supported on Windows platforms.
Printer Commands (Continued)	

Terminal Commands

Command	Description
BELL	Disables or reenables the terminal bell on warning messages.
BREAK	Enables or disables Intr (interrupt), Stop, Susp (suspend), and Break keys.
CLR	Clears the screen.
CS	Clears the screen.
GET.TERM.TYPE	Displays the terminal type.
PTERM (UNIX)	Sets and displays terminal characteristics.
PTERM (Windows Platforms)	
SET.TERM.TYPE	Sets the terminal type.
TERM	Sets printer and terminal characteristics.
WHO	Displays the port number and account name.

Terminal Commands

Tape Commands

Command	Description
ASSIGN	Assigns a device for exclusive use.
T.ATT	Assigns control of a tape drive.
T.BCK	Backs up records on a tape.
T.DET	Gives up control of the tape drive.
T.DUMP	Copies files from disk to tape.
T.EOD	Advances a tape to the end-of-data mark.
T.FWD	Advances records on a tape.
T.LOAD	Copies files from tape to disk.
T.RDLBL	Reads a tape label at the beginning of a reel.
T.READ	Reads and displays a tape record.
T.REW	Rewinds a tape to the load point.
T.SPACE	Advances a specified number of files on the tape or to the end-of-file mark.
T.UNLOAD	Rewinds and unloads the tape.
T.WEOF	Writes an end-of-file mark on the tape.
T.WTLBL	Writes a tape label in a format compatible with a Pick or REALITY system.
UNASSIGN	Releases control of a device.

Tape Commands

Managing the System

Command	Description
ABORT	Aborts the current process.
ABORT.LOGIN	Aborts the current process and runs the LOGIN paragraph.
ANALYZE.SHM	Displays information about disk and printer shared memory segments.
AVAIL	Displays statistics about disk records.
BREAK	Enables or disables Intr (interrupt), Stop, Susp (suspend), and Break keys.
CHAP	Changes priority level for tasks.
CONFIG	Displays active authorization parameters and current configurable parameter values.
CORE	Displays statistics about your current memory usage. This command is not supported on Windows platforms.
CSH	Invokes a UNIX C shell (<i>cs</i> <i>h</i>). This command is not supported on Windows platforms.
DATE	Displays the current day, date, and time on the terminal.
DATE.FORMAT	Changes the date format from American to international format.
DOS	Invokes a DOS command window. This command is not supported on UNIX systems unless you use a suitable emulator.
EDIT.CONFIG	Displays and edits UniVerse configurable parameters.
ENVIRONMENT or ENV	Sets and displays environment variables.
HELP	Gets a description of a UniVerse command or a BASIC statement or function.
JOBS	Lists active phantom processes.
LIMIT	Sets system limits.

System Commands

Command	Description
LISTME	Displays the status of everybody logged in to your account.
LISTU	Displays the status of users on the system.
LOGON	Starts up a phantom process. This command is not supported on Windows platforms.
MAIL	Invokes the UNIX mail processor. This command is not supported on Windows platforms.
MAKE.MAP.FILE	Creates a map file of the system catalog space.
MAP	Displays the contents of the system catalog space.
MESSAGE	Sends a message from your terminal to another user.
NOTIFY	Sets notification delay.
PAGE.MESSAGE	Suppresses the message <code>Press any key to continue . . .</code> at the end of paged reports.
PASSWD	Sets or changes the password for the current account.
PHANTOM	Starts a phantom process.
PORT.STATUS	Lists currently active UniVerse jobs.
PTIME	Displays system usage in the current terminal session.
SET.REMOTE.ID	Defines a user name and password to use for access to files on a remote UniVerse system.
SETPTR.DEFAULT	Changes the default values used by SETPTR over the entire system.
SH	Invokes a UNIX Bourne shell (<i>sh</i>). This command is not supported on Windows platforms.
SLEEP	Suspends a process.
STATUS	Displays information about users and resources.
TANDEM	Watches the input and output displayed on another user's terminal from your terminal. This command is not supported on Windows platforms.

System Commands (Continued)

Command	Description
TIME	Displays the current system date and time on your terminal.
UMASK	Sets default file permissions.
USERS	Displays the number of users currently logged in to the system.
uvbackup (UNIX) uvbackup (Windows Platforms)	Begins a daily, weekly, or full save of an entire system, or selected directories or files.
UVPROMPT	Changes the command-line prompt character, the select-list-active character, and the line-continue character.
uvrestore (UNIX) uvrestore (Windows Platforms)	Restores an entire system, or selected directories, files, or records in UniVerse hashed files saved by <i>uvbackup</i> .
WHO	Displays the port number and account name.
System Commands (Continued)	

Managing Locks

Command	Description
CLEAR.LOCKS	Clears task synchronization locks.
LIST.LOCKS	Displays the status of task synchronization locks.
LIST.READU	Displays the status of locked records.
LOCK	Sets task synchronization locks.
MASTER	Releases task synchronization locks.
RELEASE	Unlocks records.
SEMAPHORE.STATUS	Displays the status of system semaphores. This command is not supported on Windows platforms.
UNLOCK	Clears file, group, and READU locks.
UNLOCK SEMAPHORE	Clears system semaphores. This command is not supported on Windows platforms.

Lock Commands

Managing Transaction Logging

Command	Description
ACTLIST	Activates files for transaction logging.
CREATE.LDIR	Creates the log directory for the transaction logging system.
CREATE.LFILE	Creates log files in the transaction logging log directory.
DEACTLIST	Deactivates files for transaction logging.
DEL.RFILE	Deletes a series of log files that have been restored and rolled forward.
DELETE.LFILE	Deletes empty log files from the transaction logging log directory.
ENABLE.RECOVERY	Enables the transaction logging system.
LOGRESTORE	Restores transaction logging log files from tape to disk.
LOG.SAVE	Saves transaction logging files from disk to tape.
MKFILELIST	Creates a list for files to activate for transaction logging.
RECOVERY.CHECKPOINT	Finds the numbers of the first and last log files needed for a rollforward recovery.
RECOVERY.CONSISTENT	Clears the flag that indicates a file is in an inconsistent state.
RELEASE.LFILE	Releases a transaction logging log file for reuse.
SET.LOGATTR	Sets or changes the transaction logging operating mode.
SHUTDOWN.RECOVERY	Disables the transaction logging system.
SUSPEND.RECOVERY	Suspends the transaction logging system.

Transaction Logging Commands

NLS Commands

Command	Description
GET.FILE.MAP	Displays the name of the map associated with a file.
GET.LOCALE	Displays the current locale settings in NLS mode.
LIST.LOCALES	Displays information about available NLS locales.
LIST.MAPS	Displays information about available NLS maps.
NLS.UPDATE.ACCOUNT	Updates the contents of your VOC file for UniVerse in NLS mode.
RESTORE.LOCALE	Restores a previously saved locale setting.
SAVE.LOCALE	Saves the current locale setting.
SET.FILE.MAP	Associates a map with a file.
SET.GCI.MAP	Sets a global map for all GCI operations in NLS mode or displays the GCI map setting.
SET.LOCALE	Disables a locale or sets a new locale.
SET.SEQ.MAP	Assigns maps to UNIX files or pipes or Windows NT files opened with the BASIC OPENSEQ or OPENDEV statement.
UNICODE.FILE	Converts a mapped file to an unmapped file without making a copy of the file. This command also converts an unmapped file to a mapped file.

NLS Commands

Index

Numerics

1NF keyword 2-17
32BIT keyword 2-17
64BIT keyword 2-18

A

.A command 1-11
A keyword 2-18
ABORT command 1-34
ABORT.LOGIN command 1-35
account commands A-17
accounts, conversion of 1-128
ACCOUNT.FILE.STATS
 command 1-36, 4-22, 4-26
ACTLIST command 1-39
ADDING keyword 2-18
AFTER keyword 2-18
ALL keyword 2-18
ALL.MATCH keyword 2-19
ANALYZE.FILE command 1-41
ANALYZE.SHM command 1-55
AND keyword 2-19
ANY keyword 2-19
APPEND keyword 2-20
APP.PROGS file 4-16
ARCHIVE keyword 2-20
ARE keyword 2-20
arithmetic commands A-16
AS keyword 2-20
-AS keyword 2-20
ASSIGN command 1-60
ASSOC keyword 2-21
ASSOC.WITH keyword 2-21
asterisk (*) command 1-29

AT keyword 2-22
-AT keyword 2-22
AUTOLOGOUT command 1-63
AUTONL keyword 2-22
AUX.MAP keyword 2-22
AUX.PORT keyword 2-22
AVAIL command 1-64
AVERAGE keyword 2-13, 2-22
AVG keyword 2-23

B

BANNER keyword 2-23
BASE keyword 2-24
BASIC command 1-66
BASIC keyword 2-24
BASIC programs, managing A-15
BCI keyword 2-24
BEFORE keyword 2-24
BELL command 1-69
bill-of-materials records 2-108
BLK keyword 2-24
BLOCK keyword 2-24
BLOCK.PRINT command 1-70, 4-17
BLOCK.TERM command 1-72, 4-17
BLTRS file 4-17
BREAK command 1-74
BREAK keyword 2-25
BREAK-ON keyword 2-25
BREAK.ON keyword 2-13, 2-25
BREAK.SUP keyword 2-13, 2-27
BRIEF keyword 2-27
BUILD.INDEX command 1-75
BY keyword 2-28
BY-DSND keyword 2-28
BY-EXP keyword 2-29

BY-EXP-DSND keyword 2-31
 BY.DSND keyword 2-28
 BY.EXP keyword 2-29
 BY.EXP.DSND keyword 2-31

C

.C command 1-13
 CALC keyword 2-13, 2-33
 CALCULATE keyword 2-33
 CANCEL keyword 2-33
 –CANCEL keyword 2-34
 case-sensitivity, data types 1-127
 CATALOG command 1-77, 4-9
 CATALOG keyword 2-34
 catalog space 4-9
 CD command 1-83
 CENTURY.PIVOT command 1-85
 CHANGE.DOMAIN command 1-88
 CHAP command 1-87
 CHDIR command 1-90
 CHECKPOINT keyword 2-34
 CHECK.SUM command 1-91
 Circus database 1-371, 1-372, 1-471, 1-472
 CLEAN.ACCOUNT command 1-94, 4-13, 4-14
 CLEAR keyword 2-34
 CLEARCOMMON command 1-101
 CLEARDATA command 1-102
 CLEARPROMPTS command 1-103
 CLEARSELECT command 1-104
 CLEAR.FILE command 1-97
 CLEAR.LOCKS command 1-99
 CLR command 1-105
 CNAME command 1-106
 COL-HDR-SUPP keyword 2-35
 COLLATE keyword 2-36
 COL-SUPP keyword 2-35
 COL.HDG keyword 2-34
 COL.HDR.SUPP keyword 2-35
 COL.SPACES keyword 2-35
 COL.SPCS keyword 2-35
 COL.SUP keyword 2-36
 commands
 arithmetic A-16
 for managing accounts A-17
 for managing BASIC programs A-15

for managing locks A-29
 for managing menus A-21
 for managing records A-11
 for managing secondary indexes A-14
 for managing the system A-26
 for managing UniVerse files A-10, A-12
 NLS A-31
 in paragraphs A-5
 printer A-22
 quick reference A-1 to A-31
 for redirecting output A-4
 RetriVe A-6
 select lists A-8
 sentence stack A-3
 tape A-25
 terminal A-24
 transaction logging A-30
 VOC file A-19
 COMO command 1-109, 4-3
 COMPILE.DICT command 1-111
 COMPILE.DICTS command 1-113
 COMPLETE keyword 2-36
 CONCURRENT keyword 2-36
 CONFIG command 1-114
 CONFIGURE.FILE command 1-117
 CONNECT command 1-122
 options 1-122
 output from SELECT commands 1-126
 transaction statements 1-127
 CONV keyword 2-36
 conventions, documentation 1-xxiv
 converting accounts 1-128
 CONVERT.ACCOUNT command 1-128
 CONVERT.SQL command 1-129
 CONVERT.VOC command 1-140
 COPIES keyword 2-37
 –COPIES keyword 2-37
 COPY command 1-142
 COPY.LIST command 1-147
 CORE command 1-149
 COUNT command 1-150
 COUNT.SUP keyword 2-37
 CP command 1-152
 CREATE keyword 2-37
 CREATE.FILE command 1-156, 4-28

CREATE.INDEX command 1-165
 CREATE.LDIR command 1-169
 CREATE.LFILE command 1-171
 CRT keyword 2-37
 CS command 1-173
 CSH command 1-174
 CT command 1-176
 CTYPE keyword 2-38

D

.D command 1-16
 DATA
 keyword 2-38
 statement 1-178
 DATE command 1-180
 DATE.FORMAT command 1-181
 DBL-SPC keyword 2-38
 DBL.SPC keyword 2-38
 DEACTLIST command 1-185
 DECATALOG command 1-186
 DEFAULT keyword 2-38
 DEFAULTS keyword 2-39
 DEFER keyword 2-39
 DEFINE.DF command 1-191, 4-10
 DELETE command 1-199
 DELETE keyword 2-39
 DELETE.CATALOG command 1-201, 4-9
 DELETE.FILE command 1-205
 DELETE.INDEX command 1-208
 DELETE.LFILE command 1-209
 DELETE.LIST command 1-211, 4-12
 DELETING keyword 2-39
 DEL.RFILE command 1-197
 DETAIL keyword 2-40
 DET-SUPP keyword 2-39
 DET.SUP keyword 2-40
 DEVICE keyword 2-40
 DEVICELIST keyword 2-40
 devices, setting maps for 4-5
 DICT keyword 2-40
 DICT.DICT file 4-18
 DICT.PICK file 4-19
 DIFF keyword 2-41
 DISABLE LOCK.HIST keyword 2-41
 DISABLE.INDEX command 1-213
 DISKS keyword 2-41

DISPLAY command 1-215
 DISPLAY keyword 2-41
 display width, *see* columns
 displaying
 column attributes 1-126
 last value for fields 2-13, 2-14
 minimum value for fields 2-13
 multivalues 1-124
 DISPLAY.NAME keyword 2-42
 DIVERT.OUT command 1-216
 DLIST command 1-219
 documentation conventions 1-xxiv
 DOS command 1-221
 DOWN keyword 2-42
 DYNAMIC keyword 2-42

E

ED command 1-222
 EDIT.CONFIG command 1-223
 EDIT.LIST command 1-226
 EJECT keyword 2-42
 EMPTY.NULL keyword 2-42
 ENABLE LOCK.HIST keyword 2-42
 ENABLE.INDEX command 1-228
 ENABLE.RECOVERY command 1-229
 ENDPAGE keyword 2-43
 ENUM keyword 2-43
 ENUMERATE keyword 2-43
 ENVIRONMENT command 1-236
 EQ keyword 2-44
 EQUAL keyword 2-44
 ESEARCH command 1-238
 EVAL keyword 2-44
 EVALUATE keyword 2-45
 EVERY keyword 2-45
 EXCHANGE command 1-239
 EXPLODE keyword 2-45
 EXTERNAL keyword 2-45

F

FANCY.FORMAT command 1-240
 field modifiers 2-13
 field qualifiers 2-14
 FILE keyword 2-46
 file types 1-157

FILELOCK keyword 2-46
 file-managing commands A-10, A-12
 FILEMAP keyword 2-46
 files
 APP.PROGS 4-16
 BLTRS 4-17
 DICT.DICT 4-18
 DICT.PICK 4-19
 NEWACC 4-20
 NLS.MAP.DESCS 1-225
 REVISE.PROCESSES 4-21
 sp.config 1-641
 STAT.FILE 4-22
 SYS.HELP 4-23
 SYS.MESSAGE 4-24
 UNIVERSE.MENU.FILE 4-25
 UNIVERSE.STAT.FILE 4-26
 UNIVERSE.VOCLIB 4-27
 USER.HELP 4-28
 usplot 1-641
 &COMO& 4-3
 &DEVICE& 4-4
 &ED& 4-6
 &HOLD& 4-8
 &MAP& 4-9
 &PARTFILES& 4-10
 &PH& 4-11
 &SAVEDLISTS& 4-12
 &TEMP& 4-13
 &UFD& 4-14
 &UNICODE.FILE& 4-15
 FILE.OFF keyword 2-46
 FILE.ON keyword 2-46
 FILE.STAT command 1-241
 FILE.USAGE command 1-243
 FILE.USAGE.CLEAR command 1-246
 FILE.USAGE.OFF command 1-247
 FIRST keyword 2-46
 FIX keyword 2-47
 FMT keyword 2-47
 FOOTER keyword 2-47
 FOOTING keyword 2-48
 FOR keyword 2-50
 FORCE keyword 2-50
 FORM keyword 2-50
 -FORM keyword 2-50
 FORMAT command 1-250
 FORMAT.CONV command 1-252

FORMAT.MAP keyword 2-51
 FORM.FEED keyword 2-51
 -FORM.FEED keyword 2-51
 FORM.LIST command 1-248
 FROM keyword 2-51
 FTN keyword 2-51
 FUNDAMENTAL keyword 2-52

G

GE keyword 2-52
 GEN keyword 2-52
 GENERAL keyword 2-52
 GET.FILE.MAP command 1-258
 GET.LIST command 1-260, 4-12
 GET.LOCALE command 1-262, 1-391, 1-498
 GET.STACK command 1-264
 GET.TERM.TYPE command 1-265
 GO command 1-268
 GRAND-TOTAL keyword 2-52
 GRAND.TOTAL keyword 2-52
 GREATER keyword 2-53
 GROUP keyword 2-53
 GROULOCK keyword 2-54
 GROUP.SIZE keyword 2-53
 GROUP.STAT command 1-272
 GROUP.STAT.DETAIL command 1-274
 GT keyword 2-54

H

HASH.AID command 1-276
 HASH.HELP command 1-280
 HASH.HELP.DETAIL command 1-282
 HASH.TEST command 1-284
 HASH.TEST.DETAIL command 1-286
 HDR-SUPP keyword 2-54
 HDR.SUP keyword 2-54
 -HEAD keyword 2-54
 HEADER keyword 2-54
 HEADING keyword 2-55
 HELP command 1-289, 1-677, 4-23, 4-28
 HEX keyword 2-57

-HEX keyword 2-57
 HOLD keyword 2-57
 HUSH command 1-292
 HUSH keyword 2-57

I

.I command 1-18
 IAM command 1-293
 ID-SUP keyword 2-58
 ID-SUPP keyword 2-58
 ID.ONLY keyword 2-58
 ID.SUP keyword 2-58
 IF command 1-295
 IN keyword 2-58
 IN2 keyword 2-58
 IN2.FORMAT keyword 2-59
 INFO keyword 2-59
 INFORM keyword 2-59
 INFORMATION keyword 2-59
 INFORMATION.FORMAT
 keyword 2-59
 INITIALIZE.CATALOG command 1-
 297
 INITIALIZE.DEMO command 1-299
 in-line prompting (<<...>>)
 command 1-31
 INODE keyword 2-59
 INPLACE keyword 2-60
 INQUIRING keyword 2-60
 INTERNAL keyword 2-60
 INTERSECT keyword 2-60
 INTR.KEY X record 3-4
 inversion, *see* case inversion
 INVERT keyword 2-60
 INVISIBLE keyword 2-60
 ISOLATION keyword 2-61
 IS.NOT.NULL keyword 2-61
 IS.NULL keyword 2-61

J

JOBS command 1-300
 JOIN.BUFFER keyword 2-61

K

KEEP keyword 2-62
 KEEP.COMMON keyword 2-62
 keyword symbols 2-12

L

.L command 1-19
 labels 1-268
 LARGE.RECORD keyword 2-62
 LAYER.STACK keyword 2-62
 LE keyword 2-63
 LENGTH keyword 2-63
 LESS keyword 2-63
 licensing 1-679
 LIKE keyword 2-63
 LIMIT command 1-301
 LIST command 1-302
 LIST keyword 2-63
 -LIST keyword 2-63
 LISTDOS command 1-351
 LISTF command 1-351
 LISTFL command 1-351
 LISTFR command 1-351
 LISTK command 1-351
 LISTM command 1-351
 LISTME command 1-347
 LISTO command 1-351
 LISTPA command 1-351
 LISTPH command 1-351
 LISTPQ command 1-351
 LISTR command 1-351
 LISTS command 1-351
 LISTSL command 1-351
 LISTU command 1-349
 LISTUN command 1-351
 LISTV command 1-351
 LIST.DF command 1-306
 LIST.DIFF command 1-307
 LIST.FILE.STATS command 1-313,
 4-22, 4-26
 LIST.INDEX command 1-315
 LIST.INTER command 1-319
 LIST.ITEM command 1-322
 LIST.LABEL command 1-325
 LIST.LOCALES command 1-330
 LIST.LOCKS command 1-332

LIST.MAPS command 1-334
 LIST.READU command 1-336
 LIST.SICA command 1-341
 LIST.UNION command 1-344
 LNUM keyword 2-63
 local commands, prefix character 1-
 125
 LOCAL keyword 2-64
 LOCK command 1-353
 lock-managing commands A-29
 LOCKS keyword 2-64
 LOCK.HIST keyword 2-64
 LOCK.WAIT keyword 2-64
 LOGIN paragraph 1-359
 LOGON command 1-361
 LOGOUT command 1-362
 LOGTO command 1-364
 LOGTO.ABORT command 1-365
 LOG.RESTORE command 1-355
 LOG.SAVE command 1-357
 LOOP command 1-366
 lowercase, *see* case inversion, case-
 sensitivity
 LPTR keyword 2-64
 LT keyword 2-65

M

MAIL command 1-368
 MAKE.DEMO.FILES command 1-
 371
 MAKE.DEMO.TABLES command 1-
 372
 MAKE.MAP.FILE command 1-373,
 4-9
 managing
 BASIC programs A-15
 locks A-29
 menus A-21
 records A-11
 the system A-26
 transaction logging A-30
 MAP
 command 1-375
 keyword 2-65
 maps and devices 4-5
 MARGIN keyword 2-65
 MASTER command 1-376

MATCHES keyword 2-65
 MATCHING keyword 2-65
 MAX keyword 2-13, 2-68
 ME keyword 2-69
 MENUS command 1-379
 menus, managing A-21
 MENU.DOC command 1-377
 MENU.PIX command 1-378
 MERGE.LIST command 1-380
 MERGE.LOAD keyword 2-69
 MESSAGE command 1-383
 MFILE.HIST keyword 2-69
 MIN keyword 2-13, 2-69
 MINIMIZE.SPACE keyword 2-70
 MINIMUM.MODULUS keyword 2-71
 MKFILELIST command 1-385
 –MODIFY keyword 2-71
 modulo 1-157
 MONETARY keyword 2-71
 MOTIF command 1-387
 MTU keyword 2-71
 multiple SELECT statements 1-127
 multivalues, displaying 1-124
 MULTI.VALUE keyword 2-72
 MVDISPLAY keyword 2-73
 MVDISPLAY option 1-124

N

NE keyword 2-73
 NEEDNL keyword 2-73
 NETWORK keyword 2-73
 NEWACC files 4-20
 NEWACC keyword 2-73
 NEW.PAGE keyword 2-73
 NEXT keyword 2-73
 NEXT.AVAILABLE keyword 2-74
 NFMT keyword 2-74
 NHEAD keyword 2-74
 NLSDEFGCIMAP parameter 1-501
 NLS.ADMIN command 1-390
 NLS.MAP.DESCS file 1-225
 NLS.UPDATE.ACCOUNT
 command 1-391
 NO keyword 2-74
 NODE keyword 2-76
 NODEFAULT keyword 2-76

NOEJECT keyword 2-77
 NOFMT keyword 2-77
 NOHEAD keyword 2-77
 –NOHEAD keyword 2-77
 NOHOLD keyword 2-77
 NOKEEP keyword 2-77
 NONE keyword 2-77
 NOPAGE keyword 2-78
 NORETAIN keyword 2-78
 NOT keyword 2-78
 NOTIFY command 1-393
 NOT.MATCHING keyword 2-78
 NOXREF keyword 2-80
 NO.INDEX keyword 2-75
 NO.MATCH keyword 2-75
 NO.NEW keyword 2-75
 NO.NULLS keyword 2-75
 NO.PAGE keyword 2-75
 NO.SELECT keyword 2-75
 NO.SPLIT keyword 2-76
 NO.WAIT keyword 2-76
 NO.WARN keyword 2-76
 NSELECT command 1-394
 NULL keyword 2-80
 NUMERIC keyword 2-81
 NUM.SUP keyword 2-81

O

ODBC.CONNECTIONS keyword 2-81
 OF keyword 2-81
 OFF
 command 1-396
 keyword 2-81
 ON keyword 2-81
 ONLY keyword 2-82
 ON.ABORT paragraph 1-398
 ON.EXIT paragraph 1-399
 OPTIM.SCAN keyword 2-82
 OR keyword 2-82
 OS keyword 2-82
 output
 redirecting A-4
 from SELECT commands 1-126
 OVERWRITING keyword 2-82

P

PAGE.MESSAGE command 1-402
 paragraph commands A-5
 parentheses as keyword 2-17
 PASSWD command 1-403
 PCT keyword 2-14, 2-82
 PDICT keyword 2-83
 PERCENT keyword 2-14, 2-83
 PERCENTAGE keyword 2-83
 PERCENT.GROWTH keyword 2-83
 PHANTOM command 1-404, 4-11
 phantom processes 4-11
 PICK keyword 2-84
 PICK.FORMAT keyword 2-84
 PID keyword 2-84
 POPEN keyword 2-84
 PORT keyword 2-85
 –PORTNO keyword 2-85
 PORT.STATUS command 1-407
 prefix character 1-125
 PREFIX keyword 2-85
 PRIME command 1-411
 PRINT keyword 2-85
 printer commands A-22
 printer driver scripts 1-527
 PRINTER keyword 2-85
 –PRINTER keyword 2-85
 PRINT.ADMIN command 1-412
 PRIORITY keyword 2-85
 procedures 1-127
 PROGRAMSIZE keyword 2-86
 programs, unsupported 4-16
 PROMPT keyword 2-87
 PTERM command 1-413, 1-430
 PTIME command 1-438
 P.ATT command 1-400
 P.DET command 1-401

Q

QSELECT command 1-439
 question mark (?) command 1-28
 quick reference
 UniVerse commands A-1 to A-11
 UniVerse keywords 2-11
 QUIT command 1-441
 QUIT.KEY X record 3-6

R

.R command 1-21
 RADIX command 1-442
 RAID command 1-446
 RAID.GUI command 1-449
 –RANGE keyword 2-87
 READLOCK keyword 2-87
 READULOCK keyword 2-87
 REALITY keyword 2-87
 REALITY.FORMAT keyword 2-87
 REBUILD.DF command 1-452
 RECORD command 1-456
 RECORD keyword 2-88
 records, managing A-11
 RECORD.SIZE keyword 2-88
 RECOVERY.CHECKPOINT
 command 1-460
 RECOVERY.CONSISTENT
 command 1-462
 redirecting output A-4
 REFORMAT command 1-463
 –REJECT keyword 2-88
 relational operators 2-12
 RELEASE command 1-468
 RELEASE.LFILE command 1-469
 RELLEVEL X record 3-8
 REMOVE.DEMO.TABLES
 command 1-472
 REMOVING keyword 2-89
 REPEAT command 1-366, 1-473
 report qualifiers 2-15
 REPORTING keyword 2-89
 QUEUE keyword 2-89
 REQUIRE.INDEX keyword 2-89
 REQUIRE.SELECT keyword 2-89
 RESIZE command 1-474
 RESTORE keyword 2-90
 RESTOREDATA keyword 2-90
 RESTORE.LOCALE command 1-479
 RETAIN keyword 2-90
 Retrieve commands A-6
 ReVise 4-21
 REVISE command 1-480
 REVISE.PROCESSES file 4-21
 RUN command 1-483
 RUNNING keyword 2-90

S

.S command 1-23
 SAID keyword 2-90
 SAMPLE keyword 2-91
 SAMPLED keyword 2-91
 SAVEDATA keyword 2-91
 SAVE.LIST command 1-485, 4-12
 SAVE.LOCALE command 1-487
 SAVE.STACK command 1-488
 SAVING keyword 2-91
 SCHEMA keyword 2-92
 SCHEMAS keyword 2-92
 SEARCH command 1-489
 secondary index commands A-14
 SELECT command 1-492
 select list commands A-8
 SELECT.BUFFER keyword 2-92
 SELECT.ONLY keyword 2-92
 SEMAPHORE.STATUS command 1-495
 sentence stack commands A-3
 separation 1-157
 SEQ.NUM keyword 2-93
 SET keyword 2-93
 SETFILE command 1-521
 SETPTR command 1-522, 1-530, 4-8
 sending options to printer driver
 scripts 1-527
 SETPTR.DEFAULT command 1-539
 setting maps for devices 4-5
 SETUP.DEMO.SCHEMA
 command 1-540
 SET.FILE command 1-497
 SET.FILE.MAP command 1-498
 SET.GCI.MAP command 1-501
 SET.INDEX command 1-503
 SET.LOCALE command 1-507
 SET.LOG.ATTR command 1-509
 SET.REMOTE.ID command 1-511
 SET.SEQ.MAP command 1-513
 SET.SQL command 1-515
 SET.TERM.TYPE command 1-518
 SH command 1-541
 SHOW keyword 2-93
 SHUTDOWN.RECOVERY
 command 1-543
 SINGLE.VALUE keyword 2-93
 SLEEP command 1-545

SOME keyword 2-93
 SORT command 1-547
 SORT.ITEM command 1-551
 SORT.LABEL command 1-554
 SPLIT.LOAD keyword 2-94
 SPOKEN keyword 2-94
 SPOOL command 1-573, 1-576
 SPOOL keyword 2-94
 –SPOOL keyword 2-94
 SP.ASSIGN command 1-559, 1-561
sp.config file 1-641
 SP.EDIT command 1-562, 1-567
 SP.TAPE command 1-571
 SQL keyword 2-94
 SQUAWK keyword 2-95
 SREFORMAT command 1-578
 SSELECT command 1-583
 STACKWRITE X record 3-9
 STARTPAGE keyword 2-95
 STAT command 1-586
 STATISTICS keyword 2-95
 STATS keyword 2-96
 –STATUS 2-96
 STATUS command 1-588
 –STATUS keyword 2-96
 STAT.FILE file 4-22
 stored procedures, *see* procedures
 storing unsupported programs 4-16
 subrecords 2-108
 SUM command 1-589
 SUPP keyword 2-96
 –SUPPRESS keyword 2-96
 SUSPENDED keyword 2-96
 SUSPEND.RECOVERY command 1-593
 SUSP.KEY X record 3-10
 SYBASE, case-sensitivity 1-127
 symbols for keywords 2-12
 SYSTEM keyword 2-97
 system, managing A-26
 SYS.HELP file 4-23
 SYS.MESSAGE file 4-24

T

TABLE keyword 2-97
 TANDEM command 1-616
 tape commands A-25

TAPE keyword 2-97
 TEMPL keyword 2-97
 TERM command 1-619
 terminal commands A-24
terminfo file 1-520
 TEST keyword 2-97
 THAN keyword 2-97
 THE keyword 2-97
 THEN keyword 1-295, 2-98
 TIME command 1-622
 TIME keyword 2-98
 TO keyword 2-98
 TOTAL keyword 2-14, 2-99
 transaction logging commands A-30
 transactions, statements 1-127
 TRANSPORT keyword 2-14
 TRAP keyword 2-100
 TRUNCATE keyword 2-100
 TTY.OFF keyword 2-100
 TTY.ON keyword 2-100
 TYPE keyword 2-101
 T.ATT command 1-594
 T.BCK command 1-596
 T.DET command 1-597
 T.DUMP command 1-598
 T.EOD command 1-601
 T.FWD command 1-602
 T.LOAD command 1-603
 T.RDLBL command 1-605
 T.READ command 1-607
 T.REW command 1-609
 T.SPACE command 1-610
 T.UNLOAD command 1-611
 T.WEOF command 1-612
 T.WTLBL command 1-613

U

.U command 1-25
 UMASK command 1-623
 UNASSIGN command 1-626
 UNICODE keyword 2-101
 –UNICODE keyword 2-101
 UNICODE map name 1-513
 UNICODE.FILE command 1-627
 UNION keyword 2-101
 UNIQUE keyword 2-101
 UniVerse

licensing 1-679
 UNIVERSE.MENU.FILE file 4-25
 UNIVERSE.STAT.FILE file 4-26
 UNIVERSE.VOCLIB file 4-27
 UNLIKE keyword 2-102
 UNLOCK command 1-629
 unsupported programs 4-16
 UP keyword 2-102
 UPDATE.ACCOUNT command 1-633
 UPDATE.INDEX command 1-635
 UPDATING keyword 2-102
 uppercase, *see* case inversion, case-sensitivity
usa command 1-637
usd command 1-641
 USER keyword 2-102
 user records 3-2 to 3-11
 USERS
 command 1-642
 keyword 2-102
 USER.HELP file 4-28
 USING keyword 2-102
usm command 1-643
usp command 1-645
usplug file 1-641
uv command 1-647
uvbackup command 1-653, 1-657
 UVFIXFILE command 1-669
uvfixfile command 1-662
uvlictool command 1-679
 UVPROMPT command 1-680
uvrestore command 1-681, 1-685
 UV.LOGIN paragraph 1-649
 UV.VI command 1-651

V

VCATALOG command 1-689
 VERIFIED keyword 2-103
 VERIFILE keyword 2-103
 VERIFY keyword 2-104
 VERIFY.DF command 1-691
 VERIFY.SQL command 1-693
 VERT keyword 2-105
 VERTICALLY keyword 2-105
 VI command 1-699
 VLIST command 1-701

VOC file commands A-19
 VVOC command 1-703

W

WHEN keyword 2-105
 WHO command 1-707, 4-12
 width of display columns
 WITH keyword 2-106, 2-107, 2-108
 WITHIN keyword 2-108, 2-109

X

.X command 1-26
 X records 3-2 to 3-11
 X-descriptors 3-2
 –XREF keyword 2-109

Symbols

? (question mark) command 1-28
 .A command 1-11
 .C command 1-13
 .D command 1-16
 .I command 1-18
 .L command 1-19
 .R command 1-21
 .S command 1-23
 .U command 1-25
 .X command 1-26
 (...) keyword 2-17
 @ASSOCROW keyword 2-17
 * (asterisk) command 1-29
 &COMO& file 4-3
 &DEVICE& file 4-4, 4-5
 &ED& file 4-6
 &HOLD& file 4-8
 &MAP& file 4-9
 &PARTFILES& file 4-10
 &PH& file 4-11
 &SAVEDLISTS& file 4-12
 &TEMP& file 4-13
 &UFD& file 4-14
 &UNICODE.FILE& file 4-15
 <<...>> (in-line prompt) command 1-31