Rocket® software

Rocket UniVerse

User Reference

Version 11.3.5

January 2023
UNV-1135-USRR-1

Notices

Edition

Publication date: January 2023
Book number: UNV-1135-USRR-1
Product version: Version 11.3.5

Copyright

© Rocket Software, Inc. or its affiliates 1985-2023. All Rights Reserved.

Trademarks

Rocket is a registered trademark of Rocket Software, Inc. For a list of Rocket registered trademarks go to: www.rocketsoftware.com/about/legal. All other products or services mentioned in this document may be covered by the trademarks, service marks, or product names of their respective owners.

Examples

This information might contain examples of data and reports. The examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

License agreement

This software and the associated documentation are proprietary and confidential to Rocket Software, Inc. or its affiliates, are furnished under license, and may be used and copied only in accordance with the terms of such license.

Note: This product may contain encryption technology. Many countries prohibit or restrict the use, import, or export of encryption technologies, and current use, import, and export regulations should be followed when exporting this product.

Corporate information

Rocket Software, Inc. develops enterprise infrastructure products in four key areas: storage, networks, and compliance; database servers and tools; business information and analytics; and application development, integration, and modernization.

Website: www.rocketsoftware.com

Rocket Global Headquarters
77 4th Avenue, Suite 100
Waltham, MA 02451-1468
USA

To contact Rocket Software by telephone for any reason, including obtaining pre-sales information and technical support, use one of the following telephone numbers.

Country	Toll-free telephone number
United States	1-855-577-4323
Australia	1-800-823-405
Belgium	0800-266-65
Canada	1-855-577-4323
China	400-120-9242
France	08-05-08-05-62
Germany	0800-180-0882
Italy	800-878-295
Japan	0800-170-5464
Netherlands	0-800-022-2961
New Zealand	0800-003210
South Africa	0-800-980-818
United Kingdom	0800-520-0439

Contacting Technical Support

The Rocket Community is the primary method of obtaining support. If you have current support and maintenance agreements with Rocket Software, you can access the Rocket Community and report a problem, download an update, or read answers to FAQs. To log in to the Rocket Community or to request a Rocket Community account, go to www.rocketsoftware.com/support.

In addition to using the Rocket Community to obtain support, you can use one of the telephone numbers that are listed above or send an email to support@rocketsoftware.com.

Contents

Chapter 1: UniVerse commands

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.

UniVerse also provides system-level commands, which you execute from the shell prompt. System-level commands are stored in the $UVBIN directory.

.A

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

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.

Syntax

.A [*n*] *text*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
n	The line number of the sentence in the stack to which you want to append text. If you do not include the line number, *text* is appended to sentence 1.
text	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 *text*.

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.

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

Syntax

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

Parameters

The following table describes each parameter of the syntax.

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

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.

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

Syntax

.D [*n* | *name*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
n	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.
name	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.

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

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.

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.

Syntax

.L [*n* | *name*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
n	The number of sentences to display. The number can be from 1 through 99.
name	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.

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.

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.

Syntax

.R [*n* | *name*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
n	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.
name	The name of the sentence or paragraph. The named sentence or paragraph is inserted in the sentence stack.

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.

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.

Syntax

.S *name* [*start* [*end*]]

Parameters

The following table describes each parameter of the syntax.

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

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
n	Specifies the number of the sentence to be converted to uppercase. If you do not specify *n*, sentence 1 is converted.

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.

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

Syntax

.X [*n*]

Parameters

The following table describes each parameter of the syntax.

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

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.

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.

Syntax

?

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.

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.

Syntax

***** [*comment*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
comment	The text of the comment.

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.

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.

Syntax

```
<< [control,] … text [, option] >>
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description	
control	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*n*	Uses word *n* from the command line as the input value. (The verb is word 1.)
	F(*fn*,*rn* [,*fm* [,*vm* [,*sm*]]])	Finds the input value from a record (*rn*) in a file (*fn*). You must specify the file and the record. Separate them with a comma (no spaces allowed). *vm* extracts a value, and *sm* extracts a subvalue from the field (*fm*). Separate values with commas (no spaces allowed).
	I*n*	Takes word *n* from the command line, but prompts if word *n* was not entered.
	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(*string*)	Repeats the prompt until the user presses ENTER, and inserts *string* between each entry.
	S*n*	Takes word *n* 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.
	@(*col*, *row*)	Prompts at this column and row number on the terminal.
text	The prompt to be displayed.	
option	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.	

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] [VERBOSE]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
filenames	Specifies the files whose statistics you want to gather. If a select list of file names is active, you do not need to specify *filenames*.
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.
VERBOSE	Displays the file names being processed on the screen.

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

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 STAT.FILE 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.

Example(s)

Example 1

The STAT.FILE has been enhanced to indicate the number of bits in each file (32 or 64), and the revision of each file. The following example shows a STAT.FILE:

```
  >SORT STAT.FILE FILENAME FILETYPE FILEMOD FILESEP BIT REV ID.SUP

 File Name.........    Type    Mod.....    Sep    Bit    Rev.

 D_&SAVEDLISTS&           3           1      2     32      12
 &SAVEDLISTS&             1                        DIR     DIR
 D_VOC                   12           3      2     32      12
 VOC                     17          47      4     32      12
 D_VOCLIB                 3           1      2     32      12
 VOCLIB                   2           7      4     32      12
```

Example 2

Starting with UniVerse v11.3.3 or v12.2.1:

Additional fields have been added to the STAT.FILE file to help find which files should be resized first after running the ACCOUNT.FILE.STATS command. The additional fields are:

- GRP.OVR.100
- PCT.GRP.OVR.100
- FILESIZE.MB
- FILESIZE.GB

If you use one of the following phrases, then the files can be sorted with the problem files first:

- PCT.GRP.OVR.MB

- PCT.GRP.OVR.GB
- PCT.GRP.OVR (copy of PCT.GRP.OVR.MB)

```
>ACCOUNT.FILE.STATS ALL LOCAL
>LIST STAT.FILE PCT.GRP.OVR

LIST STAT.FILE PCT.GRP.OVR 11:12:44am  21 Apr 2021  PAGE    1
ID..................... File Name.......... Bit Mod..... %Grp>100% File.......
                                                          Size(MB)...

19470*40354.6154*0016   SALES                32      101    101.00       45.03
19470*40354.6154*0020   VOC                  32        3      3.00        0.04
19470*40354.6154*0017   D_TESTFILE           32        1      1.00        0.00
19470*40354.6154*0018   TESTFILE             32   100000      0.00      439.45
19470*40354.6154*0022   VOCLIB               32        7      0.00        0.02
19470*40354.6154*0019   D_VOC                32        3      0.00        0.00
```

You can see from the **%Grp>100%** column that the SALES and VOC data files, and the dictionary for TESTFILE, should be resized first.

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
key.id	The key ID to activate.
password	The password corresponding to *key.id*.
ON *hostname*	The name of the remote host on which you want to activate the encryption key.

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.

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.

Syntax

ACTLIST *listname*

Parameters

The following table describes each parameter of the syntax.

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

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.

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, the style, the revision, 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.

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.
filename	The name of the file.
NO.PAGE	Suppresses automatic paging.
LPTR	Sends output to the printer via logical print channel *n*.
STATISTICS	Lists additional information about dynamic files.
STATS	Same as STATISTICS.

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
ORDERSFile name ..................  ORDERS
 Pathname ...................        ORDERS
 File type ..................        DYNAMIC
 File style and revision ....        64BIT Revision 12
 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
 File style and revision ....        64BIT Revision 12
 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
```

```
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 UniVerse shared memory structure. Use `analyze.shm` from an operating system command prompt.

> **Note:** `analyze.shm` is the same command as the smat command. `smat` is shorthand for "Shared Memory Analysis Tool" and is used at the operating system level.

Use the ANALYZE.SHM command in the UniVerse environment.

Syntax

`analyze.shm` {*options*}

Parameters

You must specify at least one option and precede each option with a hyphen. Options are case sensitive and must be preceded with a hyphen. For example -r shows record locks where -R shows U2 Data Replication status.

Parameter	Description
-a [*userno*]	(All) Lists information generated by all options. *userno* specifies a user number. You must be UniVerse Administrator to use the *userno* 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.
-n	(Numbers) Lists raw, unformatted data for all table entries, including unused entries.
-o	(Not supported for AIX) Lists only the OpenSSL version currently used by Universe.
-p [*userno*]	Display shared memory information for the specified user number. Requires UniVerse Administrator privileges.
-r	(Readu) Lists all active record locks. The UniVerse `LIST.READU` command also lists this information.
-R	(Replication) Lists information about the state of U2 Data Replication.

-s	(Semaphore) Lists the status of system semaphores. The `SEMAPHORE.STATUS` command also lists this information.
	
	Starting at 11.3.1, when U2 Audit Logging is enabled, -s shows audit logging daemon semaphores. The following example shows the semaphores for an audit of `uvaudd`:
	
	`ready 0`
	`mlock 1 owner 0`
	
	The following example shows the semaphores for an audit of buffer 0:
	
	`mutex 1 owner 0`
	`fillOK 1`
	`useOK 0`
	`active 0`
	
	The first column is the semaphore name; the second column is the current value of the semaphore.

-t [0]	(Tunables) Lists current configurable parameter values. 0 suppresses display of the asterisk that indicates the parameter has been changed from the default.
-u	(User) Lists the status of task synchronization locks. The UniVerse `LIST.LOCKS` command also lists this information.

Parameter	Description

-v	You can change the SYNCALOC and UVSYNC `uvconfig` parameters while UniVerse is running. Changes will take effect immediately without having to restart UniVerse. To do this, use the `analyze.shm` command with the -v option. Any parameter values that are changed will be displayed as shown in the following example:
	
	`analyze.shm -v "UVSYNC=0" "SYNCALOC=0"`
	`UVSYNC = 0`
	
	In the previous example, UVSYNC was changed from 1 to 0 so the new value was displayed. Because SYNCALOC was already set to 0, no change was made and a new value was not reported.
	
	Parameter changes made in this way are temporary and only apply to the currently running UniVerse run time and are discarded when UniVerse is stopped. To make the changes permanent, the `uvconfig` file must be updated.

-V	You can change the SYNCALOC and UVSYNC `uvconfig` parameters while UniVerse is running. Changes will take effect immediately without having to restart UniVerse. If you want to put the parameters to change in a file with the new values in a file, use the -V option. The following example illustrates a configuration file showing that the parameters and their associated values must be space delimited. Use of "=" as an assignment operator is not allowed in this context.
	
	`UVSYNC 1`
	`SYNCALOC 0`
	
	To apply the changes, use the following command:
	
	`bin/analyze.shm -V` *`confparams`*
	`SYNCALOC = 0`
	`UVSYNC = 0`
	
	This will generate output for each parameter that is changed, however if a parameter is not changed, then it will not be displayed, similar to the -v option.
	
	Parameter changes made in this way are temporary and only apply to the currently running UniVerse run time and are discarded when UniVerse is stopped. To make the changes permanent, the `uvconfig` file must be updated.

-w	Only lists processes waiting for file locks or record locks.
-x	Lists general system information not displayed by the other options.
-z	Includes network licensed users, when used with -c and -x.

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
```

User sessions

The -p option shows the user session information 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 user settings.

The following command displays a memory map showing used and unused sections for a user's session. 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.Pinchrs: 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
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
```

U2 Data Replication

The next example shows information about the state of U2 Data 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 UniVerse shared memory structure.

Use the analyze.shm or smat command from an operating system command prompt to examine the disk and user sessions.

Syntax

`ANALYZE.SHM` {*options*}

Parameters

You must specify at least one option and precede each option with a hyphen. Options are case sensitive and must be preceded with a hyphen. For example -r shows record locks where -R shows U2 Data Replication status.

Parameter	Description
-a [*userno*]	(All) Lists information generated by all options. *userno* specifies a user number. You must be UniVerse Administrator to use the *userno* 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.
-n	(Numbers) Lists raw, unformatted data for all table entries, including unused entries.

Parameter	Description
-p [*userno*]	Display shared memory information for the specified user number. Requires UniVerse Administrator privileges.
-r	(Readu) Lists all active record locks. The UniVerse `LIST.READU` command also lists this information.
-R	(Replication) Lists information about the state of U2 Data Replication.

-s	(Semaphore) Lists the status of system semaphores. The `SEMAPHORE.STATUS` command also lists this information.
	
	Starting at 11.3.1, when U2 Audit Logging is enabled, -s shows audit logging daemon semaphores. The following example shows the semaphores for an audit of `uvaudd`:
	
	`ready 0`
	`mlock 1 owner 0`
	
	The following example shows the semaphores for an audit of buffer 0:
	
	`mutex 1 owner 0`
	`fillOK 1`
	`useOK 0`
	`active 0`
	
	The first column is the semaphore name; the second column is the current value of the semaphore.

-t [0]	(Tunables) Lists current configurable parameter values. 0 suppresses display of the asterisk that indicates the parameter has been changed from the default.
-u	(User) Lists the status of task synchronization locks. The UniVerse `LIST.LOCKS` command also lists this information.
-x	Lists general system information not displayed by the other options.
-z	Includes network licensed users, when used with -c and -x.

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
```

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
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
```

U2 Data Replication

The next example shows information about the state of U2 Data 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	
device	Specifies the device you want to assign. *device* 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 *n*	Specifies the logical tape unit. *n* 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 *n*	Specifies a logical print channel. *n* 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 *mapname*	Specifies that you want to set a map for the device. *mapname* must be built and installed in shared memory. *mapname* 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.
	DEFAULT	Specifies *mapname* as the name in the NLSDEFDEVMAP parameter in the `uvconfig` 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 *size*	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.	

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*]

Parameters

The following table describes each parameter of the syntax.

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

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*]

Parameters

The following table describes each parameter of the syntax.

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

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.

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.

Syntax

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

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
filename	The name of a type 1 or type 19 file containing BASIC programs. Filenames can include only ASCII characters, no multibyte characters.
programs	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 *programs*.
*	Specifies all programs in the file.

The following table describes available options.

Option	Description
+$*option*	Turns on the specified $OPTIONS option. *option* can also be the name of a UniVerse flavor. See the `$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.
-$*option*	Turns off the specified $OPTIONS option. *option* can also be the name of a UniVerse flavor. See the `$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.
-I	Suppresses execution of `RAID` or `VLIST` on a compiler or a BASIC program.
-L	Generates a listing of the program.
-LIST	Same as -L.
-X	Generates a cross-reference table of statement labels and variable names used in the program.
-XREF	Same as -X.
-S	Generates a listing of the program and spools it directly to the printer rather than to a file.
-SPOOL	Same as -S.
-T	Suppresses the symbol and line number tables that are usually appended to the end of an object file for run-time error messages.

File name extension support

The `BASIC` command supports file name extensions for BASIC source and object code. For the command `BASIC BP MYPROGRAM`, the `BASIC` command will look first in the BP file for a file called `MYPROGRAM`.

If it does not find `MYPROGRAM`, it will look for `MYPROGRAM.IBAS`. If it finds `MYPROGRAM.IBAS`, it compiles `MYPROGRAM.IBAS` and places the object code in the same file and call it `MYPROGRAM.IRUN`. This behavior is enabled by setting *PI_BASIC* to 1 in `uvconfig` file and running `uvregen`.

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.

BLOCK.PRINT

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

Syntax

BLOCK.PRINT *string* [*string* …]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
string	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.

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     BBBBBBBBBBB          CCCCCCCCCC
    AAAAAA     BBBBBBBBBBBB     CCCC     CCCCCCCC
  AAAAAAAA     BBBB      BBBB   CCCC           CCCC
 AAAA   AAAA   BBBBBBBBBBBB     CCCC           CCCC
AAAA    AAAA   BBBBBBBBBBB      CCCC
AAAAAAAAAAAA   BBBB      BBBB   CCCC           CCCC
AAAAAAAAAAAA   BBBB      BBBB   CCCC           CCCC
AAAA     AAAA  BBBBBBBBBBBB       CCCCCCCCCCCC
AAAA      AAAA  BBBBBBBBBBB        CCCCCCCCCC
```

BLOCK.TERM

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

Syntax

BLOCK.TERM *string* [*string* …]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
string	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.

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     BBBBBBBBBBB          CCCCCCCCCC
    AAAAAA     BBBBBBBBBBBB       CCCCCCCC CCCC
   AAAAAAAA    BBBB      BBBB    CCCC        CCCC
  AAAA   AAAA   BBBBBBBBBBBB     CCCC          CCCC
 AAAA     AAAA  BBBBBBBBBBB      CCCC
 AAAAAAAAAAAA   BBBB      BBBB   CCCC            CCCC
 AAAAAAAAAAAA   BBBB      BBBB   CCCC            CCCC
 AAAA     AAAA  BBBBBBBBBBBB       CCCCCCCCCCCC
 AAAA      AAAA  BBBBBBBBBBB          CCCCCCCCCC
  >
```

blook

Use the `blook` command to detect corruption in an alternate index.

> **Note:** This command is reserved for use by U2 engineering and support and is included in this manual for reference purposes only.

Syntax

blook [*options*] *filename*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
options	The option to use with `blook` command. See the following table for descriptions of the `blook` options.
filename	The name of the file to want to run `blook` against.

Options

The following table describes the options available with the `blook` command.

Option	Description
+a *address*	Displays the nodes from the address you specify to the end of the file. Follow the +a with the address where you want the scan to begin. You should specify a valid node/leaf address. All node/leafs are 8192 bytes in size.
-a *address*	The -a option limits the examination of the index to a single leaf/node. You should specify a valid node/leaf address. All node/leafs are 8192 bytes in size.
-d	The -d option displays both the keys and the data in a leaf or node. Keys and data refer to internal information. For example, the data displayed from an index lead may include information used to maintain the sort order of the B-tree as well as the actual user data.
-k	The -k option extracts the keys within the B-tree. When the B-tree is an index to a data file, keys refer to the indexed value rather than the record keys in the data file
-i *keylist*	The -i option locates a specific key in the index. *keylist* is the indexed value, not the key, to an item in the data file. This option can be used with the -d option to display data associated with a specific indexed value. For example, `blook -d -i SMITH INDEX.000` displays the data record keys for all items indexed as "SMITH."
-n	The -n option disables automatic pagination.
-r	The -r option displays the data in descending order rather than ascending order.
-s	The -s option displays statistics about the B-tree, including index data and oversized nodes or leafs.
-v	The -v option verifies file integrity and displays summary information.
-V	The -V option verifies file integrity and displays detailed information about the index.
-p *key*, *pass*	The -p option activates an encryption key with the password you specify. If you do not specify the password, UniVerse returns "Unable to decrypt the data: You must provide password to access the key."

You can use the LINES environment variable to control the display depth and the COLS environment variable to control the display width.

BREAK

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

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

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.

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] [BRIEF] [CONCURRENT] [RESET]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
filename	The name of a UniVerse file for which to build a secondary index. If *filename* is not specified, BUILD.INDEX prompts for it.
indexes	A list of fields in *filename* 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.
BRIEF	Suppresses a validation to see if the dictionary item has changed since the index was last built. To suppress screen output from BUILD.INDEX use the HUSH ON/OFF commands.
CONCURRENT	Specifies to build the index online. Building the index online is slower than building the index offline, but users can make updates to the file while UniVerse is rebuilding the index.
RESET	Use the RESET keyword if BUILD.INDEX CONCURRENT was interrupted before it completed. The RESET keyword changes the build state of the index from “Building” to “Required.”

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, unless you specify the CONCURRENT option. Building the index concurrently is slower than building it offline, but you can make updates to the file which UniVerse is rebuilding the index.

If you build an index using the CONCURRENT option, UniVerse creates an INDEX.log file in the INDEX.log directory to log the status of the build process. The following example illustrates the INDEX.log file contents:

```
# pg l_BIGFILE/INDEX.log
2011/03/29 07:49:22 - Building index started (for "F1")
2011/03/29 07:55:35 - 10% finished
2011/03/29 08:07:14 - 20% finished
2011/03/29 08:23:50 - 30% finished
2011/03/29 08:45:07 - 40% finished
2011/03/29 09:11:02 - 50% finished
2011/03/29 09:41:55 - 60% finished
2011/03/29 10:17:10 - 70% finished
(EOF):
```

Considerations when using the CONCURRENT option

Building an index using the CONCURRENT option is slower than that of a standard BUILD.INDEX process.

If a process already has the index open when it is being rebuilt, creating a new index on the file or disabling or enabling index updates on the file are not recognized by that process. Because of this, the following rules apply:

- If a new index is added, updates done by the processes which already had the file open will NOT be written to the new index.
- If index updates are re-enabled while users have the file open, updates done by these processes will NOT be written to the index.
- If a BUILD.INDEX CONCURRENT operation is run with active users not having the current state of the index file loaded, as in the previous two examples, the data file and index file are very likely to be out of sync when the operation completes.

Care should be taken to make sure all users have the most current status of the index before starting a CONCURRENT build. If a CONCURRENT rebuild of an existing index is being done, users with the file open will already have the current status of the indexes. But if a new index is being created, all users should close and reopen the file so they have the current status of the index loaded prior to the start of any BUILD.INDEX CONCURRENT process. This will ensure that all updates performed by those users during the build process are accurately reflected in the index file.

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.

>
```

CASE

The CASE command turns case sensitivity on or off for TCL commands. To turn case sensitivity off, enter CASE OFF. To turn case sensitivity on, enter CASE ON. If you enter CASE with no options, UniVerse displays the current status. CASE ON is the default.

The following example shows the behavior of UniVerse when case sensitivity is ON:

```
>CASE ON
TCL Case sensitivity is ON
>list voc f1
RetrieVe: syntax error. Unexpected sentence without filename. Token was "".
Scanned command was LIST 'voc' 'f1'
>
```

The next example shows the behavior of UniVerse when case sensitivity is OFF:

```
>CASE OFF
TCL Case sensitivity is OFF
>list voc f1
LIST VOC F1 10:53:37am 31 Oct 2013 PAGE 1
VOC......... F1.............
GLOBAL.CATDI File - Used to
R access system
catalog space.
ISOLATION Keyword -
SET.SQL
environment
MONETARY Keyword - NLS
locale category
OPTIM.SCAN Keyword -
SET.SQL
Environment
.
.
.
```

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.

Beginning at UniVerse 11.2.3, the ICATALOG command is available. The ICATALOG command is not flavor-dependent, and it is recommended that users use the ICATALOG command instead of the CATALOG command when needing to globally catalog a program.

Syntax

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

The following table describes each parameter of the syntax.

Parameter	Description
filename	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 *filename*, you cannot specify other qualifiers on the command line. In this case CATALOG prompts you for the qualifier values, one at a time.
catalog.name	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 *catalog.name*, *program.name* is the name the program will have in the catalog.
program.name	The name of the program in *filename* 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 *filename*.

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

Option	Description
FORCE	Specifies that CATALOG should overwrite any cataloged program with the same *catalog.name*. 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 *catalog.name* 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 *catalog.name*.

Option	Description
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.

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.

File name extension support

The `CATALOG` command supports a file name extension for BASIC object code.

For the command `CATALOG BP MYPROGRAM`, the `CATALOG` command will look first in the `BP.O` file for a file called `MYPROGRAM` and catalog that. If it does not find `MYPROGRAM`, it will look in the `BP` file for a file called `MYPROGRAM.IRUN` and catalog that as `MYPROGRAM`.

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.

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.

Syntax

CD *filename* [*descriptors*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
filename	The name of the UniVerse file whose dictionary entries are to be compiled. Do not specify the DICT keyword.

Parameter	Description
descriptors	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.

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 system-wide 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
year	A four-digit year. The first two digits specify the century, the last two digits specify the pivot year.
nn	A two-digit code indicating a sliding pivot year. *nn* can be 00 through 99.

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 system-wide 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 though 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.

CHANGE.ENCRYPTION.PASSWORD

Use the `CHANGE.ENRYPTION.PASSWORD` command to change the password for an encryption key.

Syntax

`CHANGE.ENCRYPTION.PASSWORD` `ID` *`existing_password new_password`* `[NOCASCADE]`

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
ID	The name of the encryption key or the wallet ID.
existing_password	The current password for the encryption key or wallet.
new_password	The new password for the encryption key or wallet.
NOCASCADE	By default, when you change the password for a key, the password for that key in all wallets that contain that key is also changed. You can choose not to change the password in wallets by specifying the NOCASCADE option.

The new password should conform to password policies. To specify no password, enter a quoted empty string (“”) on the command line. If you do not specify the `CHANGE.ENCRYPION.PASSWORD` parameters on the command line, UniVerse prompts you for the current password and the new password. To specify no password, press ENTER when prompted.

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.

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.

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.

Syntax

CHANGE.DOMAIN *domainnew.domain*

Parameters

The following table describes each parameter of the syntax.

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

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.

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.

Syntax

CHDIR *pathname*

Parameters

The following table describes each parameter of the syntax.

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

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 *filename*. If you do not specify DICT, records in the data file are checked.
USING [DICT];*dictname*	*If DICT is not specified, uses dictname*; as the dictionary of *filename*. If DICT is specified, the dictionary of *dictname*; is used as the dictionary of *filename*.
filename	The file whose records you want to checksum. You can specify *filename* ;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.
records	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 *n*	Specifies the records whose record ;IDs are stored in select list *n*.
selection	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.
field	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.
report.qualifiers	One or more of the following keywords: ▪ FIRST ▪ LPTR ▪ ONLY ▪ SAMPLED ▪ ID.ONLY ▪ NOPAGE ▪ SAMPLE These keywords modify the report format.

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
field	The name of the field for which the statistics are produced.
filename	The name of the file for which the statistics are produced.
bytes	The total number of bytes for values in the field.
avg.bytes	The average number of bytes of values in the field.
count	The number of records processed.
value	The checksum value for the field.
count	The bit count of the values in the field.

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.

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.

Syntax

CLEAN.ACCOUNT

&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.

Note: If a before or after DELETE trigger exists for a file, the trigger is not invoked when you use the CLEAR.FILE command.

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.

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 *filename*. DATA is the default.
filename	Specifies the name of the file to be cleared. If you do not specify *filename* and there is no active select list, CLEAR.FILE prompts for the file name. If you do not specify *filename* and there is an active select list, CLEAR.FILE clears the files named by the select list. If you specify *filename* and there is an active select list, CLEAR.FILE clears *filename* and the files named in the select list.

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.

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.

Syntax

CLEAR.LOCKS [*n*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
n	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.

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.

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.

Syntax

CLEARCOMMON

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.

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.

Syntax

CLEARDATA

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.

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.

Syntax

CLEARPROMPTS

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.

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.

Syntax

CLEARSELECT [*list#* | ALL]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
list#	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.

Example

This example clears select list 5:

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

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

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.
filename	A name of a file. *old.filename* is the filename to be changed, *new.filename* is the new name of the file.
old	The name of the record you want to change.
new	The new name of the record.

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 *record* [HUSH]	Creates a COMO record and starts copying terminal output to it. HUSH suppresses terminal display. Output is copied to record 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 {*record* | *}	Deletes a COMO record. Asterisk (*) clears the &COMO& file.
LIST	Lists COMO records in your account.
SPOOL *record* [T]	Prints a COMO record on the line printer. T displays a COMO record on the screen.

COMO is short for **com**mand **o**utput.

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 OFFis 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
LISTCOMO 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.

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.

Syntax

COMPILE.DICT *filename* [*descriptors*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
filename	The name of the UniVerse file whose dictionary entries are to be compiled. Do not specify the DICT keyword.
descriptors	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.

Example

This example compiles four I-descriptors:

```
>COMPILE.DICT
ORDERSCompiling "*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.

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.

Syntax

COMPILE.DICTS

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.

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
               SCRMAX =           5
               SCRSIZE=           512
               QDEPTH =           16
               HISTSTK=           99
               QSRUNSZ=           2000
               QSBRNCH=           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
               PKRJUST =       0
               PROCACMD=       0
               PROCRCMD=       0
               PROCPRMT=       0
               ALLOWNFS=       0
               CSHDISPATCH             =       /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
Current using OpenSSL version:
                    LIBCRYPTO = /nome4/uv113/uvssl/libcrypto.so.1.2
                    LIBSSL = /nome4/uv113/uvssl/libssl.so.1.2
                    OPENSSL = /nome4/uv113/uvssl/openssl
```

CONFIGURE.FILE

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

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.

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.
filename	The name of the dynamic file to be reconfigured.

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

Parameter	Description
MINIMUM.MODULUS *n*	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 *n*	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 *n*	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%.
LARGE.RECORD *n*	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.

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.

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.

Syntax

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

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
data.source	The name of the data source to which you want to connect. The data source must be defined in the uvodbc.config file. If you do not enter the name of a data source, CONNECT lists all data sources in the uvodbc.config file.
option	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 *setting*.
setting	The new setting for the option.

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.
string	Enables block mode (see ON). *string* must be from one to four characters. To terminate the block of SQL statements, enter *string* immediately after an “SQL+”prompt.

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.

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).
char	Displays a value mark as *char* (one character).

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).
string	Displays SQL null as *string*. The string can be from one to four characters. By default, null is displayed as the four-character string NULL.

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:

Character	Description	Character	Description
!	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

UVOUT option

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

Value	Description
filename	Stores output in *filename* on the client, then displays the output from *filename*. 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.

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.

WIDTH option

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

Value	Description
col#,width	Sets the width of column *col#* to *width*. Do not enter a space after the comma. Specify *col#* as an asterisk (*) to set the width of all columns. *width* 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 *width*.
F	Folds data that is wider than the specified *width* onto multiple lines.
?	Displays the current column width settings, and tells whether data will be truncated or folded.

CONVERT.ACCOUNT

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

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*.

Syntax

CONVERT . ACCOUNT

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.

`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.

Syntax

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

Parameters

The following table describes each parameter of the syntax.

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

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 *options* with the CREATE action.
INFO	Lists D-, A-, and S-descriptors in the dictionary of *filename*.
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 *options* except SAVEDATA with the TEST action.

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 *filename*_SQLSAVE. Use this option only with the CREATE action.
LPTR	Sends output to the printer.

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.

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).
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.
   - *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,*s*) where *s* 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

This conversion code...	Generates this SQL date type...
BB or BX	VARBIT
All others	VARCHAR

- 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 (*n*) or VARCHAR(*n*) where *n* is the field's width.

@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	
C*tns*/*old*/*new*[/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.	
	t	C or empty for a column, A for an association, or K for key part.
	n	1-, 2-, or 3-digit column, association, or key part number.
	s	Optional spaces.
C*tns*T*stype*	Changes the data type of a column definition to *type*.	
	t	C or empty for a column, or K for key part.
	n	1-, 2-, or 3-digit column or key part number.
	s	Optional spaces.
	type	Any valid SQL data type except CHAR VARYING, CHARACTER VARYING, NCHAR VARYING, or BIT VARYING (use VARCHAR instead).
DA*n*	Deletes association *n*.	
U*tnsa*	Uses another column or association definition. *a* 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`.	
R[*tn*]	Redisplays the SQL-formatted definitions for the single column, key part, or association *n,* 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.	

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 *filename*_SQLSAVE, then executes the `CREATE TABLE` statement and returns to the UniVerse prompt.

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):

```
                            Type &
Field......... Field. 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:

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

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

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

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.

Note: Beginning at UniVerse 11.2.3, the ICOPY command is available. The ICOPY command is not flavor-dependent.

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:

rec1rec2rec3 …

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.rec1rec2* ,*new.rec2rec3* ,*new.rec3 …*

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

rec1rec2 **,***new.rec2rec3rec4* **,***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.

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.
source.file	The name of the file containing the records to copy. If you do not specify DICT, records are copied from the data file.
target.file	The name of the file to which the records are copied. If you do not specify DICT, records are copied to the data file.
rec	The record ID of the record to be copied. Separate multiple record IDs with spaces.
new.rec	The record ID of the new copy of the record. If you do not specify *new.rec*, 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 *new.rec*.
ALL	Copies all records in the source file to the target file. The ALL keyword overrides an active select list.

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 *n*	Copies the first *n* records in *source.file*. `COPY` creates select list 0 (unless a select list is active) and copies the records specified by the first *n* 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 [*n*]	Sends output to a printer via logical print channel *n*. If you do not specify *n*, logical print channel 0 is used. If you specify LPTR and CRT, `COPY` ignores CRT and sends output to the printer.

Option	Description
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.
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 *target.file*. The new record overwrites the old.

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.

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

For more information, see &SAVEDLISTS&, on page 478.

Syntax

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

Parameters

The following table describes each parameter of the syntax.

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

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.
>
```

COUNT

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

If you enter COUNT with no options, it counts and reports the total number of records in the 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 *filename*. If you do not specify DICT, records in the data file are counted.
USING [DICT] *dictname*	If DICT is not specified, uses *dictname* as the dictionary of *filename*. If DICT is specified, the dictionary of *dictname* is used as the dictionary of *filename*.
filename	Specifies the file containing the records you want counted. You can put *filename* 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 *n*	Specifies that select list *n* is to be used.
selection	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.

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.
filename	The file containing the record you want to print.
records	The record IDs of records you want to print.
*	Specifies all records in *filename*.

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 -UNICODE 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.

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
key.id	The encryption key ID.
password	The password for *key.id*.

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 passwords contains a space (" "), you must use quotation marks to enclose the password.

CREATE.ENCRYPTION.WALLET

Use the `CREATE.ENCRYPTION.WALLET` command to create and encryption wallet, which contains encryption keys and passwords.

Syntax

`CREATE.ENCRYPTION.WALLET` *`wallet_id [wallet_password]`*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
wallet_id	The encryption wallet ID.
wallet_password	The password for *key.id*.

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.

Note: Beginning at UniVerse 11.2.3, the `ICREATE` command is available. The `ICREATE` command is not flavor-dependent, and it is recommended that users use the `ICREATE` command instead of the `CREATE` command, when needing to globally catalog a program.

`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 *NLS Guide*.

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: type : 3 modulo : 1 separation: 2
DATA	Specifies only the data file.
filename	The UniVerse filename. CREATE.FILE creates the data file with the name *filename* and the file dictionary with the name D_*filename*. If you do not specify *filename*, CREATE.FILE prompts for it. *filename* must follow the UniVerse naming conventions. For more information, see File naming conventions.
datafile	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.
type	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 *type*, CREATE.FILE prompts for it.
modulo	An integer value greater than zero (limited to 8,388,608 in UniVerse versions prior to 11.1.0) defining the number of groups in the file. If you do not specify *modulo*, CREATE.FILE prompts for *modulo* and *separation*. *modulo* is ignored for nonhashed and dynamic files.
separation	An integer from 1 through 8,388,608 specifying the group buffer size in 512-byte blocks. If you do not specify *separation* and one is needed, the file is created with a separation of 4, unless you are in prompting mode. If you press Return in response to the *separation* prompt, the file is not created. *separation* is ignored for nonhashed and dynamic files.
description	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 *filename* by entering ?filename. You must specify *description* in quotation marks when you create dynamic files.

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.
GROUP.SIZE *n*	Specifies the size of each group in the file, where *n* is the separation, which is a multiple of 2K of the value entered. The default is 1 for a separation of 2K.
LARGE.RECORD *n*	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%.
MERGE.LOAD *n*	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%.
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.
MINIMUM.MODULUS *n*	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.
RECORD.SIZE *n*	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.
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.
SPLIT.LOAD *n*	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%.

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.

File naming conventions

UniVerse file names 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)	?.

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

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 In PICK or REALITY flavor, the VOC entry appears as: LONG.OVERDUE 001 F 002 D_LONGOVERDUE 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

CREATE.INDEX

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

Syntax

```
CREATE.INDEX [DICT | USING [DICT] dictname] [filename] [AT account]
[fields] [ALL.NULLS] [NO.NULLS] [TO RELATIVE.PATH] [BRIEF] [COLLATE {
locale | OFF}]
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies a file dictionary.
USING *dictname*	If DICT is not specified, uses *dictname* as the dictionary of *filename*. If DICT is specified, the dictionary of *dictname* is used as the dictionary of *filename.*
filename	The name of the file containing fields you want to index. If you do not specify *filename*, `CREATE.INDEX` prompts you for it.
account	Specifies the UniVerse account where you want to store the indexes you create. *account* must be defined in the UV.ACCOUNT file. If *filename* already has indexes, you cannot use the AT clause because all indexes are stored as files in one account directory.
fields	A list of fields separated by spaces to use as secondary access keys. If *fields* is not specified, `CREATE.INDEX` prompts you to enter a single field.
ALL.NULLS	Specifies that the index will not contain any data. This option would be used when creating an index based on an I-type subroutine acting as a trigger. After creation, the index does not require a build. Executing `BUILD.INDEX` on an ALL.NULLS index will return a message indicating that a build is not required. **Note:** Indexes created with the ALL.NULLS keyword are not compatible with releases that do not support the ALL.NULLS keyword.
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.
TO RELATIVE.PATH	Sets the index location to be a path relative to the current directory. This is useful when you copy or move accounts from one location to another, as the path to the index file does not need to be changed.
BRIEF	Suppresses the screen output.
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.

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 T*file* 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 T*file* 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.

Note: If you resize a file from 32-bit to 64-bit and the file contains an index, the index will remain a 32-bit file. You must delete and re-recreate the index to change it to a 64-bit file.

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.

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.

Syntax

CREATE.LDIR *pathname*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
pathname	The absolute path of the log directory.

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.

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

lg*nnn*

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.

Syntax

CREATE.LFILE *sizefiles*

Parameters

The following table describes each parameter of the syntax.

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

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.

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.

Syntax

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

Parameters

The following table describes each parameter of the syntax.

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

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 06 11:43 I_CREDITORS
-rw-rw-r-- 1 susan acctg 1536 Jul 06 11:43 CREDITORS
-rw-rw-r-- 1 susan acctg 2048 Jul 06 11:42 D_CREDITORS
-rw-r--r-- 1 susan acctg 49152 Jul 06 11:41 VOC
drwxrwxrwx 2 susan acctg 512 Jul 02 16:55 I_ORDERS
-rw-r--r-- 1 susan acctg 3072 Jul 02 16:55 D_ORDERS
drwxrwxr-x 2 susan acctg 512 Jul 02 16:43 LONG.OVERDUE
-rw-rw-r-- 1 susan acctg 4096 Jul 02 16:41 D_LONG.OVERDUE
-rw-rw-r-- 1 susan acctg 2048 Jul 02 16:40 D_OVERDUE
-rw-rw-r-- 1 susan acctg 1536 Jul 02 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.
filename	The file containing the record you want to display.
records	The record IDs of records you want to display.
*	Specifies all records in the file.

options can be any of the following:

Option	Description
-FORM.FEED	Lists each record on a separate page.

Option	Description
-HEX	Copies data in hexadecimal format. You can use this option whether NLS mode is on or off. You cannot use -HEX with the -UNICODE 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.

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.

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.

Syntax

DATA *data*

Parameters

The following table describes each parameter of the syntax.

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

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.

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*.

Syntax

DATE

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.
conversion	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).

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.ENCRYPTION.KEY

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

Syntax

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

Parameters

The following table describes each parameter of the syntax.

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

DEACTLIST

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

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*.

Syntax

DEACTLIST *listname*

Parameter

The following table describes each parameter of the syntax.

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

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
filename	The name of the program file. File names can include only ASCII characters, no multibyte characters. If you do not specify *filename*, you are prompted for it.
program	The name of the program to be decataloged. Program names can include only ASCII characters, no multibyte characters. If you do not specify *program*, DECATALOG prompts for it. If there is an active select list, it is used for *program*.

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 TESTTEST removed from VOC.
 Object DELETEd from BP.O.
```

DECRYPT.FILE

The DECRYPT.FILE command decrypts data 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*]]...

Parameters

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	
filename	The UniVerse file name. If you do not specify *filename*, `DECRYPT.FILE` prompts for the name. *filename* must follow the UniVerse naming conventions. For more information about naming conventions, see File naming conventions.	
type	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.	
modulo	An integer value greater than zero (limited to 8,388,608 in UniVerse versions prior to 11.1.0) defining the number of groups in the file you are decrypting. UniVerse ignores *modulo* if you specify a nonhashed or dynamic file type.	
separation	An integer from 1 through 8,388,608 specifying the group buffer size in 512-byte blocks. If you do not specify *separation* and one is needed, the file is created with a separation of 4, unless you are in prompting mode. If you press Return in response to the *separation* prompt, the file is not created. *separation* is ignored for nonhashed and dynamic files.	
30	Decrypts a dynamic file.	
dynamic	Decrypts a dynamic file.	
USING *partition*	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 /u4: `>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.	
fieldname,key,pass	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.	
	fieldname	The name of the field to decrypt. Beginning at UniVerse 11.1, you can specify the @ID as a field name.
	key	The key ID to use for the field decryption.
	pass	The password corresponding to the *key*.

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.
GROUP.SIZE *n*	Specifies the size of each group in the file, where *n* is the separation, which is a multiple of 2K of the value entered. The default is 1 for a separation of 2K.

Parameter	Description
LARGE.RECORD *n*	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%.
MERGE.LOAD *n*	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%.
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.
MINIMUM.MODULUS *n*	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.
RECORD.SIZE *n*	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.
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.
SPLIT.LOAD *n*	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%.

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.

DECRYPT.INDEX

Use the `DECRYPT.INDEX` command to decrypt an index file associated with a field. This command does not rebuild the index.

Syntax

```
DECRYPT.INDEX <filename> <field>[, <key>[,<pass>]]
[<field>[,<key>[,<pass>]]
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
filename	The name of the file for which you want to decrypt the index.
field	The name of the field you want to decrypt.
key	The encryption key.
pass	The password for the encryption key.

If the file has @ID or WHOLERECORD encryption, you cannot decrypt any encrypted indexes. If the data field is still encrypted, you cannot decrypt the index associated with the field.

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.

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.

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.filenamepart.namepart#* [*part.namepart#*] … [FORCE]

To change the partitioning algorithm of a distributed file:

DEFINE.DF [DATA] *dist.filenamealgorithm* [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.
dist.filename	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.
part.name	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.
part#	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 *part#*.
algorithm	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 for more information. **Note:** You must specify the algorithm when adding the first part file.
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.

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.

Note: You must specify the algorithm when adding the first part file to a distributed file.

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] filename] record.ID | expression}`	
	DATA	Specifies that the algorithm expression is stored in the data file of *filename*. If you do not use the DATA keyword, the expression is stored in the dictionary of *filename*.
	filename	The name of the file containing the algorithm expression.
	record.ID	The ID of the record containing the algorithm expression. If you do not specify *filename*, the record is assumed to be an entry in the VOC file.
	expression	The I-type expression used as the partitioning algorithm. Enclose *expression* in quotation marks if it contains spaces. Fields referred to by the I-type expression must be in the same file as the record containing the I-descriptor. If you enter *expression* 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 routine` *routine* is the external routine used as the partitioning algorithm.	
SYSTEM [*s*]	Specifies the default partitioning algorithm. *s* is the character used as a separator in record IDs. If you do not specify *s*, a hyphen (-) separator is the default.	

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.

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.

Syntax

DEL.RFILE *first.log last.log* [*logdir.path*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
first.log	The number of the first log file to delete.
last.log	The number of the last log file to delete.
logdir.path	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.

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.
filename	The name of the file from which records are deleted.

Parameter	Description
records	The IDs of records to be deleted, separated by spaces. Do not specify *records* if select list 0 is active.

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*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
catalog.name	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.

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. 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.

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.
key.id	The encryption key to delete.
password	The password for the encryption key to delete.

DELETE.ENCRYPTION.WALLET

Use the `DELETE.ENCRYPTION.WALLET` command to delete an encryption wallet. You must be the owner of the file or logged on as root or a UniVerse Administrator to delete an encryption wallet. If you specify FORCE, UniVerse deletes the encryption wallet even if it is referenced by an encrypted record or field.

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.
key.id	The encryption wallet to delete.
password	The password for the encryption wallet to delete.

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.
filename	The name of the file to delete. If you do not specify *filename*, DELETE.FILE uses select list 0 if it is active. The record IDs in the select list must be UniVerse file names.

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 DECRYPT.INDEX 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.

If you do not specify either *indexes* or ALL, DELETE.INDEX prompts you to enter an index name.

Syntax

DELETE.INDEX [DICT] [*filename*] [*indexes* | ALL] [BRIEF]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.

Parameter	Description
filename	The name of a UniVerse file from which to delete a secondary index. If you do not specify *filename*, DELETE.INDEX prompts for it.
indexes	A list of index names separated by spaces. *indexes* are the names of fields in *filename* used as secondary index keys.
ALL	Specifies to delete all the secondary index keys in the file.
BRIEF	Suppresses the screen output.

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.

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.

Syntax

DELETE.LFILE *files*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
files	The number of log files you want to delete from the log directory. DELETE.LFILE deletes only files whose status is Available.

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
listname	The record ID of the saved list in the &SAVEDLISTS& file. If you do not specify *listname*, `DELETE.LIST` deletes a list called &TEMP*port#*&. See the `SAVE.LIST` command for information about this *listname*.
*	Specifies all *listnames* in the &SAVEDLISTS& file. `DELETE.LIST` clears the &SAVEDLISTS& file, leaving only your saved sentence stacks.

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.LISTSaved list "&TEMP-11057&" DELETEd.
```

DISABLE.DECRYPTION

Use the `DISABLE.DECRYPTION` command to turn off decryption on a field you specify.

Syntax

`DISABLE.DECRYPTION` *`filename <field_list>`*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
filename	The name of the file on which you want to disable decryption.
field_list	A comma-separated list of fields for which you want to disable decryption. Do not enter spaces between the field names.

DISABLE.INDEX

Use `DISABLE.INDEX` to disable automatic updating of the secondary key indexes of a file.

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 `UPATE.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.

Syntax

DISABLE.INDEX [DICT] [*filenames*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
filenames	The names of the UniVerse files for which to disable secondary index updating. If you do not specify *filenames*, DISABLE.INDEX prompts for them.

DISABLE.TANDEM

Individual UniVerse sessions can change the default TANDEM behavior. Use DISABLE.TANDEM to forbid the session to be TANDEMized.

For more information, see the TANDEM command.

Syntax

DISABLE.TANDEM [FORCE]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description

FORCE	When this option is used, DISABLE.TANDEM breaks the TANDEM connection if it is already TANDEMized. When active connection is present, but the FORCE option is not used, DISABLE.TANDEM prints the following message:
	
	`TANDEM operation is disabled.`
	`However TANDEM connection currently active`
	`(consider using FORCE option to disconnect the connection).`

DISPLAY

Use DISPLAY to display a line of text on the terminal.

Syntax

DISPLAY *text*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
text	The text that appears on the terminal.

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.

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.

Syntax

DIVERT.OUT *action* [*filenamerecord*] [*parameter*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
filename	The name of an existing type 1 or type 19 file.
record	The name of a record in *filename* where the output is to be diverted.

action can be one of the following:

Action	Description
ON	Starts sending output to specified *record* in *filename*.
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.

Action	Description
TTY.ON	Turns on terminal (tty) output display if it is disabled.

parameter can be any of the following keywords:

Parameter	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.

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
filename	The name of the file whose dictionary contains *I-descriptor*.
I-descriptor	The record ID of the I-descriptor in the dictionary of *filename*.

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.

UNIX. The DOS command is supported only if you use a suitable emulator.

Syntax

DOS [*/c command | pathname*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
command	Specifies the command to execute.
pathname	The path of a batch file or command file to execute.

Note: If you enter a command that contains a space, you must enclose the command in quotation marks.

Example

```
>DOS /c DIR
```

ED

Use ED to invoke the UniVerse Editor.

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.

Note: If you use one type of quotation mark when editing a record, such as ED VOC O'NEAL, you must surround the string by a different quotation mark, such as ED VOC "O'NEAL".

Syntax

ED [DICT] [*filename*] [*records* | *]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Edits records in the file dictionary.
filename	The name of an existing file.
records	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.

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)
1Parameter 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 Back F-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
.nn	Selects parameter *nn*.
<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.
/*pattern*/	Searches forward through the parameters for *pattern*, if any.
pattern\	Searches backward for *pattern* if any.
//	Repeats the last search forward (made with /*pattern*/). The current stored *pattern* can be found on the help screen.
\\	Repeats the last search (made with *pattern*\).

Command	Description
?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.

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&.

`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.

Syntax

EDIT.LIST [*listname*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
listname	The name of the list you want to edit. Separate multiple list names with spaces, or use an active select list.

ENABLE.ENCRYPTION

Use the `ENABLE.ENCRYPTION` command to activate encryption on specific fields in a file.

Syntax

ENABLE.ENCRYPTION *filename <field_list>*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
filename	The name of the file on which you want to enable encryption.

Parameter	Description
field_list	A comma-separated list of fields for which you want to enable encryption. Do not enter spaces between the field names.

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.

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.

Syntax

ENABLE.INDEX [DICT] [*filenames*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
filenames	The names of UniVerse files for which to enable secondary index updating. If you do not specify *filenames*, ENABLE.INDEX prompts for them.

ENABLE.TANDEM

Individual UniVerse sessions can change the default TANDEM behavior. Use ENABLE.TANDEM to allow a session to be TANDEMized.

For more information, see the TANDEM command.

Syntax

ENABLE.TANDEM

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.

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.

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. This is the default.
NO	Removes the current logging info file.
INFORM	Displays messages during the startup process.

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 encrypt the data in a UniVerse 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
filename	The UniVerse file name. If you do not specify *filename*, ENCRYPT.FILE prompts for the name. *filename* must follow the UniVerse naming conventions. For more information about naming conventions, see “File Naming Conventions” in *UniVerse User Reference*.
type	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.

Parameter	Description	
modulo	An integer value greater than zero (limited to 8,388,608 in UniVerse versions prior to 11.1.0) defining the number of groups in the file you are encrypting. UniVerse ignores *modulo* if you specify a nonhashed or dynamic file type.	
separation	An integer from 1 through 8,388,608 specifying the group buffer size in 512-byte blocks. If you do not specify *separation* and one is needed, the file is created with a separation of 4, unless you are in prompting mode. If you press Return in response to the *separation* prompt, the file is not created. *separation* is ignored for nonhashed and dynamic files.	
30	Encrypts a dynamic file.	
dynamic	Encrypts a dynamic file.	
USING *partition*	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 /u4: `>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.	
fieldname, alg, key, pass	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.	
	fieldname	The name of the field to encrypt. Beginning at UniVerse 11.1, you can specify the @ID as a field name.
	alg	The algorithm to use for encryption.
	key	The key ID to use for the field encryption.
	pass	The password corresponding to the *key*.

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.
GROUP.SIZE *n*	Specifies the size of each group in the file, where *n* is the separation, which is a multiple of 2K of the value entered. The default is 1 for a separation of 2K.
LARGE.RECORD *n*	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%.

Parameter	Description
MERGE.LOAD *n*	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%.
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.
MINIMUM.MODULUS *n*	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.
RECORD.SIZE *n*	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.
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.
SPLIT.LOAD *n*	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%.

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.

ENCRYPT.INDEX

Use the `ENCRYPT.INDEX` command to encrypt an index file associated with a field. This command does not rebuild the index.

Syntax

`ENCRYPT.INDEX` *`<filename> <field>[, <alg>, <key>[,<pass>]] [<field>[, <alg>, <key>[,<pass>]] ...`*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
filename	The name of the file for which you want to encrypt the index.
field	The name of the field you want to encrypt.

Parameter	Description
alg	A string containing the cipher name.
key	The encryption key.
pass	The password for the encryption key.

Any field you specify must already have an index created and built. If it is already encrypted as a result of the @ID, WHOLERECORD, or field encryption, you can omit the algorithm and key specifications.

ENVIRONMENT or ENV

Use `ENVIRONMENT` or `ENV` to set and display environment variables.

`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.

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 *value*.
env.variable	The name of the environment variable you want to set or clear. *env.variable* names are case-sensitive.
value	The value to which you are setting *env.variable*. *value* can be of zero length.
CLEAR	Unsets the specified environment variable. If *env.variable* is not set, the CLEAR keyword does nothing.

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/lbin:/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*]

execuv

You can use the execuv command to start multiple concurrent UniVerse background processes. You can use the command from a shell script running interactively, from a shell script running in the background, or in a crontab file to start a cron job.

Examples

The following examples demonstrate some simple methods of using the execuv command. Additional shell syntax beyond the simple examples below can also be used to redirect stdout/stderr to another file if desired.

Interactive script

```
# cat interactive.script
#!/bin/sh
nohup /u2/uv/bin/execuv "SLEEP 100" &
nohup /u2/uv/bin/execuv "SLEEP 101" &
nohup /u2/uv/bin/execuv "SLEEP 102" &

# ./interactive.script
Sending output to nohup.out
Sending output to nohup.out
Sending output to nohup.out

# ipcs -mop | grep aceb
m 177209382 0xacebfaff -rw------ root system 1 18743432 18743432
m 132124660 0xacebfb31 -rw------ root system 1 17825934 17825934
m 161484798 0xacebfb91 -rw------ root system 1 16253070 16253070
```

Three UniVerse processes with unique signatures were started.

Background script

```
# cat back.script
#!/bin/sh
```

```
/u2/uv/bin/execuv "SLEEP 100" &
/u2/uv/bin/execuv "SLEEP 101" &
/u2/uv/bin/execuv "SLEEP 102" &
# nohup ./back.script &
[1] 17825964
# Sending output to nohup.out
[1] + Done nohup ./back.script &
# ipcs -mop | grep aceb
m 178257958 0xacebfae1 --rw------- root system 1 18743462 18743462
m 133173236 0xacebfb7f --rw------- root system 1 16253088 16253088
m 162533374 0xacebfa87 --rw------- root system 1 19464404 19464404
#
```

Three UniVerse processes are started with unique signatures.

Cron jobs

In the crontab file, use `execuv` (i.e. `/u2/uv/bin/execuv`) rather than 'uv' or 'uvsh' to start UniVerse sessions via cron. This will ensure that multiple concurrent UniVerse sessions initiated by cron will have unique signatures.

EXCHANGE

Use `EXCHANGE` to switch the contents of one record with another record in a file.

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.

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.
filename	The name of the file containing the records to be switched.
record	The name of a record whose contents will be switched with the contents 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.

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.

Syntax

FANCY.FORMAT [*filename*] [*program* | *] [-LIST]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
filename	The name of the file containing the BASIC programs.
program	The name of the BASIC program. If select list 0 is active, you need not specify *program*.
*	Specifies all programs in the file.
-LIST	Prints the program on the printer.

FILE.STAT

Use FILE.STAT to get statistical information about static hashed files. FILE.STAT displays the file type, style, revision, 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.

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.

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.
filename	The name of a UniVerse hashed file.
LPTR	Sends output to the printer.

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)
File style and revision = 32BIT Revision 12
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. This command list statistics collected by sessions that have both opened and closed files activated for statistics.

Syntax

`FILE.USAGE` `[DICT ] [`*`filenames`*`]`

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
filename	A list of filenames separated by spaces. If you do not specify filenames, `FILE.USAGE` prompts for them. If a select list is active, you need not specify filename.

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.

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.

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.

`FILE.USAGE.CLEAR` takes effect only when a file is opened within a session. This command does not have an effect in gathering usage numbers in a session which already has open files. You must close and reopen the file to begin gathering statistics.

Syntax

FILE.USAGE.CLEAR [DICT] [*filenames*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
filenames	A list of filenames separated by spaces. If you do not specify filenames, `FILE.USAGE.CLEAR` prompts for them. If a select list is active, you need not specify *filenames*.

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.

`FILE.USAGE.OFF` only takes effect when a file is opened within a session. This command does not have an effect in gathering usage numbers for a session which already has open files.

Syntax

`FILE.USAGE.OFF` `[ DICT] [`*`filenames`*`]`

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
filenames	A list of file names separated by spaces for processing. If you do not specify *filenames*, `FILE.USAGE.OFF` prompts for them. If a select list is active, you need not specify *filenames*.

fixtool

The `fixtool` utility allows you to verify and repair UniVerse files. You can verify the integrity of a file without taking the file offline, but you must have exclusive access to the file to repair the file. Before running the `fixtool` utility, make a backup copy of the corrupt file at the operating-system level, then run the `fixtool` utility from the operating-system level.

Starting at UniVerse 11.3.1, the non-integrated version of this utility called `fixtool_sa` is available. `fixtool_sa` is independent from the version of UniVerse installed and can be copied and executed to verify and repair UniVerse files on similar hardware running earlier versions of UniVerse. The `fixtool_sa` utility includes a new -force option that must be used with the -fix option to repair a file on a system where UniVerse is running. Other than the -force option, the command syntax for `fixtool_sa` is identical to `fixtool`, as described in this section. However `fixtool_sa` does not support locks set by other processes. For more information about `fixtool_sa`, see the Tech Note UNV-24493.

Syntax

`fixtool` `-file` *`filename`* `[-level` *`n`*`] [-fix [-force]] [-filepath` *`path`*`] [-help] [-logging] [-logpath` *`path`*`] [-dump] [-dumppath] [-nodump] [-zero] [-start` *`n`*`] [-stop` *`n`*`]`

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
-file *filename*	Specifies the file to analyze. You must have read and write access to the file.
-level *n*	Specifies the diagnostic level. Values can be 1-10. For details about diagnostic levels, see Using fixtool with the -level option, on page 141.
-fix	Tries to fix any corruption detected in the file. You should not use this option until you have made a backup of the file. You may have to run `fixtool` with the -fix option multiple times to completely fix the file.
-force	**fixtool_sa only**. Must be used with the -fix option to repair a file on a system where UniVerse is running.
-filepath *path*	The full path of the file to analyze. Use in conjunction with the -file option.
-help	Displays a help message.
-logging	Enables `fixtool` logging. By default, UniVerse creates the `log.`*`filename`* directory, which contains a text file called LOGFILE which contains the trace information.
-logpath *path*	The full path for the log file name.
-dump	Indicates that before image should be dumped prior to repairing corrupted items.
-dumppath	Specifies the location where the temporary file used by `fixtool` should be located. Use in conjunction with the -dump option.
-nodump	Specifies to bypass creating the temporary dump directory.
-zero	Clears a group. Use with the -start or -stop option.
-start *n*	Specifies to start from the specific group you specify. *n* is the logical group number. By default, `fixtool` begins at group 1.
-stop *n*	Stops at the specific group you specify. *n* is the logical group number. By default, `fixtool` processes all groups in the file.

Examples

This section provides examples of the output from `fixtool` with various options.

fixtool with no options

The following example illustrates the output from `fixtool` when you do not specify any options:

```
C:\U2\UV\TEST>\u2\uv\bin\fixtool -file VOC
Filename: VOC
Start Diagnostics.
        0 Errors found in physical structure.
        0 Errors found in file header.
        0 Errors found in groups.
        0 Errors found in miscellaneous chains.
        0 Orphaned buffers found in file.
813 records processed.
No errors were found.
Fixtool Completed.
```

The following table describes the output from `fixtool`.

Column	Definition
Errors found in physical structure	The number of errors detected after checking specific header elements and buffer structures.
Errors found in file header	The number of errors detected in the file header, such as invalid SICA, part block pointers, and so forth.

Column	Definition
Errors found in primary groups	The number of errors detected in the primary data space, such as the DATA.30 portion of a dynamic file or the groups referenced by the modulo in a static hashed file.
Errors found in data	The number of errors found in the header structure of each record, including oversized records. UniVerse does not display this information if you use the -start or -stop options.
Orphaned buffers found in file	The number of orphaned buffers. Orphaned buffers are overflow buffers that are not linked to any primary buffer and do not exist in the freechain list.

Using fixtool with the -level option

The level option species the verbosity you want to display in the `fixtool` output. The following table describes the verbosity levels.

Verbosity level	Description
0-3	Displays only summary information.
4	Displays the record offset, the forward pointer, the backward link, and the flag word.
5	Displays the record ID and data in a multivalued array, along with the information `fixtool` displays at level 4.

The following example illustrates the output from the `fixtool` when specifying verbosity level 4.

```
C:\U2\UV\TEST>\u2\uv\bin\fixtool -file VOC -level 4
Filename: VOC
Start Diagnostics.00000b9f4 F:00000ba10 B:00000b9c4 W:0c0300000
.
.
.
Processing group: 23
00000ba10 F:00000ba2c B:00000b9e8 W:0c0300000
00000ba2c F:00000ba48 B:00000ba0c W:0c0300000
00000ba48 F:00000ba60 B:00000ba34 W:0c0300000
00000ba60 F:00000ba7c B:00000ba54 W:0c0300000
00000ba7c F:00000baa8 B:00000ba4c W:0c0300000
00000baa8 F:00000babc B:00000ba68 W:0c0100000
00000babc F:00000bae0 B:00000ba8c W:0c0300000
00000bae0 F:00000bb04 B:00000ba98 W:0c0300000
00000bb04 F:00000bb20 B:00000bafc W:0c0100000
00000bb20 F:00000bb5c B:00000bb38 W:0c0300000
00000bb5c F:00000bb80 B:00000bb04 W:0c0100000
00000bb80 F:00000bb9c B:00000bb40 W:0c0300000
00000bb9c F:00000bbb4 B:00000bb98 W:0c0300000
00000bbb4 F:00000bbd0 B:00000bb80 W:0c0300000
00000bbd0 F:00000bbe8 B:00000bbac W:0c0300000
00000bbe8 F:00000bc04 B:00000bbcc W:0c0300000
00000bc04 F:00000bc44 B:00000bba8 W:0c0300000
00000bc44 F:00000bc64 B:00000bc24 W:0c0300000
00000bc64 F:00000bca4 B:00000bc04 W:0c0300000
        0 Errors found in groups.
        0 Errors found in miscellaneous chains.
        0 Orphaned buffers found in file.
        814 records processed.
        No errors were found.
```

The following example illustrates the output from `fixtool` when specifying verbosity level 5.

```
C:\U2\UV\TEST>\u2\uv\bin\fixtool -file VOC -level 5
.
.
.
DATA: 'V¦BP.O/SETSEQMAP.B¦B¦UBN¦¦¦¦¦BP.O'
00000bb5c F:00000bb80 B:00000bb04 W:0c0100000
ID: 'NSELECT'
DATA: 'V¦NSELECT¦I¦GSKX'
00000bb80 F:00000bb9c B:00000bb40 W:0c0300000
ID: 'SUBSTRING'
DATA: 'K¦555'
00000bb9c F:00000bbb4 B:00000bb98 W:0c0300000
ID: 'UNIQUE'
DATA: 'K¦60'
00000bbb4 F:00000bbd0 B:00000bb80 W:0c0300000
ID: 'DEFAULTS'
DATA: 'K¦246'
00000bbd0 F:00000bbe8 B:00000bbac W:0c0300000
ID: 'RIGHT'
DATA: 'K¦548'
00000bbe8 F:00000bc04 B:00000bbcc W:0c0300000
ID: 'F8'
DATA: 'D¦8¦¦¦15T¦S'
00000bc04 F:00000bc44 B:00000bba8 W:0c0300000
ID: 'SETUP.DEMO.SCHEMA'
DATA: 'V¦BP.O/DEMO.SQL.B¦B¦UBN¦¦¦¦¦BP.O'
00000bc44 F:00000bc64 B:00000bc24 W:0c0300000
ID: 'PICK.FORMAT'
DATA: 'K¦211'
00000bc64 F:00000bca4 B:00000bc04 W:0c0300000
ID: 'CATALOG'
DATA: 'V¦CATALOG¦I¦BDGZ¦catalog¦INFORMATION.FORMAT'
        0 Errors found in groups.
        0 Errors found in miscellaneous chains.
        0 Orphaned buffers found in file.
        814 records processed.
        No errors were found.
Fixtool Completed.
```

`fixtool` displays the record ID and data for each record below the corresponding record header.

Using fixtool with the -logging option

If you specify the -logging option with `fixtool`, `fixtool` directs the output to the `filename` file in the `log.`*`filename`* directory in the current account. *filename* is the name of the file you specify with the -file option. If no errors are found, the log file contains only “Start” and “Complete,” and the summary is displayed on the screen. If an error is found, the error is written to the log file, as shown in the following example.

```
Filename: VOC
Start Diagnostics.
        0 Errors found in physical structure.
        0 Errors found in file header.
        0 Errors found in primary groups.
        1 Errors found in data.
        0 Other errors found in file.
888 records processed.
1 Errors on file.
```

```
Fixtool Completed.
[root@rh73 uv]# more log.VOC/LOGFILE
Start
Filename: VOC Primary Group: 1 Overflow Group: 0 Record In Group:
        1 Group Address: 0x400 Record Address: 0x400 - Forward link is an od
d address.
Complete
[root@rh73 uv]#
```

You can also redirect output from the screen to a file, as shown in the following example:

```
# bin/fixtool -file VOC -level 4 -start 1 -stop 3 >fixtool.out
```

Using fixtool with the -fix option

When you specify the -fix option with `fixtool`, `fixtool` tries to repair file corruption. `fixtool` may not be able to correct every problem it encounters because not all types of corruption can be fixed programmatically.

You may want to run `fixtool` without the -fix option first to isolate the corrupt group, then run `fixtool` with the -fix option and the -start or -stop option to reduce the time it takes to fix the file.

In the following example, `fixtool` detects a backward link problem and an incorrect padding value in group 1:

```
# bin/fixtool -file VOC -logging -start 1
Filename: VOC
Start Diagnostics.
        0 Errors found in physical structure.
        0 Errors found in file header.
        2 Errors found in file groups.
        0 Other errors found in file.
        888 records processed.
        2 Errors on file.
Fixtool Completed.
[root@rh73 uv]# more log.VOC/LOGFILE
Start
Filename: VOC Primary Group: 1 Overflow Group: 0 Record In Group:
0 Group Address: 0x400 Record Address: 0x400 - Backward link appears
to be bad.
Filename: VOC Primary Group: 1 Overflow Group: 0 Record In Group:
0 Group Address: 0x400 Record Address: 0x400 - Padding value is inco
rrect for record ID "GLOBAL.CATDIR".
Complete
```

In the next example, `fixtool` is run with the -level option to see if the chain can be followed:

```
# bin/fixtool -file VOC -level 4 -stop 1
Filename: VOC
Start Diagnostics.
        0 Errors found in physical structure.
        0 Errors found in file header.
Processing group: 584154610688589825
000000400 F:000000470 B:000000068 W:0c0300000
        2 Errors found in file groups.
        0 Other errors found in file.
        1 records processed.
        2 Errors on file.
Fixtool Completed.
```

You can now run `fixtool` with the -fix option to repair the file.

Using fixtool with the -logpath and -filepath options

You can redirect the output from `fixtool` to a different directory using the -logpath and -logging options. You must have permissions to create directories and files in the directory you specify with -logpath. The following example illustrates redirecting the log file:

```
bin/fixtool -file VOC -start 1 -logpath /u2/uv -logging
```

You can specify the -filepath option to run `fixtool` against a file that resides in a different directory, as shown in the following example:

```
# bin/fixtool -file VOC -filepath /u2/uv/test/demo1
```

Clearing one or more groups

You can use `fixtool` to clear one or more groups in a file by using the -zero option. This option enables you to clear irreparable buffers where data cannot be recovered.

The following example illustrates the results of `fixtool` when corruption was detected:

```
# bin/fixtool -file FIXTOOL.TEST -logging -start 1
Filename: FIXTOOL.TEST
Start Diagnostics.
        0 Errors found in physical structure.
        0 Errors found in file header.
        1 Errors found in file groups.
        0 Other errors found in file.
        886 records processed.
        1 Errors on file.
Fixtool Completed.

# more log.FIXTOOL.TEST/LOGFILE
Start
Filename: FIXTOOL.TEST Primary Group: 2 Overflow Group: 0 Record I
n Group: 1 Group Address: 0x600 Record Address: 0x600 - Backward link
appears to be bad.
Complete
```

The next example illustrates how to clear the group:

```
# bin/fixtool -file FIXTOOL.TEST -zero -start 2 -stop 2
Filename: FIXTOOL.TEST

Marking groups to clear.
        1 groups marked to clear.
        1 records processed.

Start Fixing.
        1 Groups cleared.

Fixtool Completed.
```

Caution: The -zero option clears all information from the target groups. Use the -zero option only if it is necessary.

fixtool locking

Note: `fixtool_sa` does not support locking.

If you run `fixtool` without the -start, -stop, or -zero options, `fixtool` sets an exclusive lock on the file it is processing. No other user can access the file while it is being processed.

If you run `fixtool` with the -start or -stop option, `fixtool` processes the file using group locks as it scans the file. This enables users to access the file and make changes while `fixtool` is running.

If a problem is detected and the -fix option is used, then a file lock will be set. If the file lock cannot be set, then `fixtool` aborts and you have to run `fixtool -fix` again. This behavior is without regard to the -start or -stop options being used. Files can be accessed by other processes during the scan phase without regard to options used.

If you specify the -zero option, the groups you specify are not traced, but the file is momentarily locked with an exclusive lock while the groups are cleared. If `fixtool` cannot obtain an exclusive lock, executing `fixtool` fails and an error message is displayed.

fnuxi

Use `fnuxi` 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: The `fnuxi` command is run from the operating system level. The `FORMAT.CONV` command does the same thing as the `fnuxi` command, but is run from TCL.

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.

Syntax

fnuxi [*options*] *pathname*

fnuxi {-export | -import | -convert [-vocname *name*] [-force] | -drop | -undo} [-name *name*] [-silent] *pathname*

Parameters

The following table describes each parameter of the syntax.

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
-m*class*	Converts a file to the format specified by machine class. *class* can be 0, 1, or 16. If you specify multiple options, you must specify the *-m* option last.
-s	(Silent) Suppresses terminal output.
-u	Converts a file to low-order byte addressing format (big-endian). The *-u* option assumes a machine class of 0 (zero) for BASIC object code, or a class of 0 or 16 for a UniVerse file. The *-u* option is the same as *-m*0.
-v	(Verbose) Displays output. This is the default.

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 *-m*1.
-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.

Use the following options only when you are converting UniVerse SQL/SICA 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 *name*.IMPORT file.
-convert	Converts imported tables to the storage format of the current machine. This option does not execute the commands in the *name*.IMPORT file.
-vocname *name*	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.
-undo	Restores imported and converted tables to the state they were in before they were converted with the *-import* option.
-name *name*	Specifies the first part of the name of the four text files generated when you use the *-export* option. If you do not specify *name*, EXPORTEDDDL is used. You cannot use the *-name* option with the *-convert* option.
-silent	Suppresses terminal output. Use only with the *-export* or *-convert* options.

UniVerse files

If you do not specify *-m*, *-u*, or *-x*, `fnuxi` converts the file to the format of the current machine.

`fnuxi` 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 11.1 of UniVerse. Use this table to identify your machine class. The UniVerse version code is on your installation tape.

Machine class	Operating system
0	IBM AIX
1	Windows Linux Solaris x86
16	HP/UX Solaris Sparc

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 `fnuxi` twice. On the source machine, use `fnuxi` 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 `fnuxi` 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 `fnuxi` 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 `fnuxi` with the *-import* option. The user who executes this command becomes the owner of the converted table. `fnuxi` 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:

```
>fnuxi CUSTOMERS
```

The next two examples convert the ORDERS data file to low-order byte addressing format:

```
>fnuxi -u ORDERS
>fnuxi -m0 ORDERS
```

The next example converts the ORDERS file to Release 6 format for a class 1 machine:

```
>fnuxi -6m1 ORDERS
```

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.

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.

Syntax

FORM.LIST [*filename*] [*record*] [TO *n*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
filename	The file containing *record*.
record	The record ID of the record whose elements are to make up the select list.
TO *n*	Creates a numbered select list. *n* specifies the list number from 0 through 10. If you omit the TO clause, FORM.LIST creates select list 0.

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.

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.

Syntax

```
FORMAT [filename] [program | *] [-LIST]
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
filename	The name of the file containing the BASIC programs.
program	The name of the BASIC program. If select list 0 is active, you need not specify *program*.
*	Specifies all programs in the file.
-LIST	Prints the program on the printer.

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: The FORMAT.CONV command is run from TCL. The fnuxi command does the same thing as the FORMAT.CONV command, but is run from the operating system level.

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.

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
-m*class*	Converts a file to the format specified by machine class. *class* can be 0, 1, or 16. If you specify multiple options, you must specify the *-m* option last.
-s	(Silent) Suppresses terminal output.
-u	Converts a file to low-order byte addressing format (big-endian). The *-u* option assumes a machine class of 0 (zero) for BASIC object code, or a class of 0 or 16 for a UniVerse file. The *-u* option is the same as *-m*0.
-v	(Verbose) Displays output. This is the default.
-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 *-m*1.
-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.

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 *name*.IMPORT file.
-convert	Converts imported tables to the storage format of the current machine. This option does not execute the commands in the *name*.IMPORT file.
-vocname *name*	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.

Option	Description
-drop	Drops tables from the source schema after they have been exported to a target schema.
-undo	Restores imported and converted tables to the state they were in before they were converted with the *-import* option.
-name *name*	Specifies the first part of the name of the four text files generated when you use the *-export* option. If you do not specify *name*, EXPORTEDDDL is used. You cannot use the *-name* option with the *-convert* option.
-silent	Suppresses terminal output. Use only with the *-export* or *-convert* options.

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.

Note: When moving UniVerse files (not SQL tables) that contain a trigger to a machine with a different byte order, the SICA portion in the header of the UniVerse file will be converted to the new byte order.

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	Operating System
0	IBM AIX
1	Windows Linux Solaris x86
16	HP/UX Solaris Sparc

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.

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.

Syntax

GET.FILE.MAP [DICT] *filename*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
filename	The name of the file, or its path, whose map name you want to display.

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.FIPS.MODE

Use `GET.FIPS.MODE` to report the current status of FIPS mode.

Syntax

GET.FIPS.MODE

This command produces the same output as `SET.FIPS.MODE INFORM`. For more information, see SET.FIPS.MODE, on page 287.

GET.LIST

Use GET.LIST to activate a saved select list.

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.

Syntax

GET.LIST [[*filename*] *listname*] [TO *list#*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
filename	The name of a file in which the list is saved. The &SAVEDLISTS& file is the default.
listname	The name of the list to be recalled. If you do not include *listname*, GET.LIST activates the list called &TEMP*port#*&, where *port#* is the one- or two-digit identifier of the terminal you are logged in on.
TO *list#*	The select list number to assign the list to. If you do not use a TO clause, GET.LIST uses select list 0.

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
LIST SUN.MEMBERS 11:07:42am 20 Oct 1995 PAGE 1
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.

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.

Syntax

GET.LOCALE [*category* | ALL]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
category	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.

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*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
listname	The name of a saved sentence stack.

Example

```
>GET.STACK MGR.STACK
```

```
Command stack "MGR.STACK" loaded
```

GET.TANDEM.STATUS

Individual UniVerse sessions can change the default TANDEM behavior. Use GET.TANDEM.STATUS to verify whether a session is TANDEMized or not.

GET.TANDEM.STATUS returns 0 if the session is not TANDEMized, or a pid if the session is TANDEMized. The pid is the process ID of the TANDEM process.

For more information, see the TANDEM command.

Syntax

GET.TANDEM.STATUS [VERBOSE]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
VERBOSE	Displays extended system output, such as: `Session is not TANDEMized.` `Session is TANDEMized by process` *nnnn*. where *nnnn* is the pid of the TANDEM process.

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]

Parameters

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.

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

Terminal Type	Description
regent25	ADDS Regent 25
vp60	ADDS Viewpoint 60
hz1410	Hazeltine 1400
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

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 uviterm-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.

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.

Syntax

GO *label* [:]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
label	The statement label. If you do not specify *label*, execution continues with the next statement. In a GO statement, the label need not end with a colon.

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 *<key.id>* [*<password>*] {PUBLIC | *grantee* {,*grantee*...}}

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
key.id	The encryption key for which you are granting another user access.
password	The password for the encryption key for which you are granting another user access.
PUBLIC	Grants access to the encryption key to all users on the system.

Parameter	Description
grantee	Grants access to the encryption key to the *grantee* you specify. *grantee* 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. **Note:** To grant access to global users or groups, you must log on as a domain user to create keys and perform the `GRANT` operation.

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.

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.

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.
filename	The name of the file.
NO.PAGE	Suppresses automatic paging.
LPTR	Sends output to the printer.

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.DETAILwhen 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.

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.

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.
filename	The name of the file.
NO.PAGE	Suppresses automatic paging.
LPTR	Sends output to the printer.

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.
filename	The name of the file.
type	A number from 2 through 18, specifying one of the file types, or *, specifying the existing file type. You can also specify *type* as a control variable for a loop.
modulo	An integer value greater than zero (limited to 8,388,608 in UniVerse versions prior to 11.1.0) defining the number of groups in the file, or *, specifying the existing modulo. You can also specify *modulo* as a control variable for a loop.
separation	An integer from 1 through 8,388,608 specifying the group buffer size in 512-byte blocks, or *, specifying the existing separation. You can also specify *separation* as a control variable for a loop.
NO.PAGE	Suppresses automatic paging.
LPTR	Sends output to the printer.

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%
     2                   1                   0
 100%        125%        150%        175%        200% full File name
   0         0          0                   0                   0    = 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%
                        5                   0       0
 100%        125%        150%        175%        200% full

  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%
  7                  0                   0                   0
 100%        125%        150%        175%        200% full
  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.

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.

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.
filename	The name of the file.
NO.PAGE	Suppresses automatic paging.
LPTR	Sends output to the printer.

Parameter	Description
PERCENT.GROWTH	Specifies that the recommendation should be based on *n*% 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.

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:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_'
         (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.

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.

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.
filename	The name of the file.
NO.PAGE	Suppresses automatic paging.
LPTR	Sends output to the printer.

Example

`HASH.HELP.DETAIL` 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:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_'
(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.
filename	The name of the file.
type	A number from 2 through 18, specifying one of the file types, or *, specifying the existing file type. You can also specify *type* as a control variable for a loop.

Parameter	Description
modulo	An integer value greater than zero (limited to 8,388,608 in UniVerse versions prior to 11.1.0) defining the number of groups in the file, or *, specifying the existing modulo. You can also specify *modulo* as a control variable for a loop.
separation	An integer from 1 through 8,388,608 specifying the group buffer size in 512-byte blocks, or *, specifying the existing separation. You can also specify *separation* as a control variable for a loop.
NO.PAGE	Suppresses automatic paging.
LPTR	Sends output to the printer.

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.
filename	The name of the file.
type	A number from 2 through 18, specifying one of the file types, or *, specifying the existing file type. You can also specify *type* as a control variable for a loop.
modulo	An integer value greater than zero (limited to 8,388,608 in UniVerse versions prior to 11.1.0) defining the number of groups in the file, or *, specifying the existing modulo. You can also specify *modulo* as a control variable for a loop.
separation	An integer from 1 through 8,388,608 specifying the group buffer size in 512-byte blocks, or *, specifying the existing separation. You can also specify *separation* as a control variable for a loop.
NO.PAGE	Suppresses automatic paging.
LPTR	Sends output to the printer.

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 *item*, CONV lists all conversion codes. If you specify a conversion code as *item*, help for that code is listed.
PICK	Specifies help for PICK, REALITY, and IN2 flavor accounts. If you do not specify *item*, PICK lists commands with PICK-flavor-specific differences. If you specify a command as *item*, help for that command is listed.
SQL	Specifies help for SQL statements. If you do not specify *item*, SQL lists all SQL statements. If you specify an SQL statement as *item*, help for that statement is listed.

Parameter	Description
BCI	Specifies help for BASIC SQL Client Interface functions. If you do not specify *item*, BCI lists all SQL Client Interface functions. If you specify a function as *item*, help for that function is listed.
helpfile	Specifies one of the system help files: BASIC.HELP BCI.HELP CONV.HELP PICK.HELP SQL.HELP SYS.HELP
item	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 *helpfile*	Specifies one of the system help files.

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.

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 T*file* conversion:

```
>HELP CONV Tfile
```

HUSH

Use HUSH to suppress all text normally sent to the screen during processing.

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.

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.

IAM

Use IAM to change the name of your account directory.

Syntax

IAM *account.dir*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
account.dir	The account directory name you want to use.

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.

ICATALOG

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.

This command is not Pick-specific and works in all flavors.

Syntax

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

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
filename	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 *filename*, you cannot specify other qualifiers on the command line. In this case ICATALOG prompts you for the qualifier values, one at a time.
catalog.name	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 *catalog.name*, *program.name* is the name the program will have in the catalog.
program.name	The name of the program in *filename* 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. ICATALOG can also read from an active select list.
*	Specifies all programs in *filename*.

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

Option	Description
FORCE	Specifies that ICATALOG should overwrite any cataloged program with the same *catalog.name*. If you do not specify FORCE and a globally cataloged program with the same name exists, ICATALOG asks for permission to overwrite it. When a normally cataloged program is cataloged, a program with the same *catalog.name* 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 *catalog.name*.
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.

If you use ICATALOG 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, ICATALOG terminates without cataloging anything.

You must compile the source code before you use the ICATALOG 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:

```
>ICATALOG 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:

```
>ICATALOG 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:

```
>

 ICATALOG 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:

```
 >ICATALOG
 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
```

ICOPY

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

This command is not Pick-specific and works in all flavors.

Syntax

```
ICOPY 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.
source.file	The name of the file containing the records to copy. If you do not specify DICT, records are copied from the data file.
target.file	The name of the file to which the records are copied. If you do not specify DICT, records are copied to the data file.
rec	The record ID of the record to be copied. Separate multiple record IDs with spaces.
new.rec	The record ID of the new copy of the record. If you do not specify *new.rec*, 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 *new.rec*.
ALL	Copies all records in the source file to the target file. The ALL keyword overrides an active select list.

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 *n*	Copies the first *n* records in *source.file*. `ICOPY` creates select list 0 (unless a select list is active) and copies the records specified by the first *n* 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 [*n*]	Sends output to a printer via logical print channel *n*. If you do not specify *n*, logical print channel 0 is used. If you specify LPTR and CRT, `ICOPY` 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.

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 *target.file*. The new record overwrites the old.

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:

rec1rec2rec3 …

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.rec1rec2 ,new.rec2rec3 ,new.rec3 …

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

rec1rec2 ,new.rec2rec3rec4 ,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:

```
>ICOPY 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:

```
>ICOPY 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.
 >>ICOPY FROM ORDERS TO ORDERS.SEPTYou 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:

```
>ICOPY FROM
STOCK.ARCH TO STOCK.H DAT,DAT_H DELETING
REPORTINGSource 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.
 >>ICOPY 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.
 >
```

ICREATE.FILE

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

This command is not Pick-specific and works in all flavors.

Syntax

```
ICREATE.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: type : 3 modulo : 1 separation : 2
DATA	Specifies only the data file.
filename	The UniVerse filename. `ICREATE.FILE` creates the data file with the name *filename* and the file dictionary with the name D_*filename*. If you do not specify *filename*, ICREATE.FILE prompts for it. *filename* must follow the UniVerse naming conventions. For more information, see the File Naming Conventions section.
datafile	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.
type	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 *type*, `ICREATE.FILE` prompts for it.
modulo	An integer value greater than zero (limited to 8,388,608 in UniVerse versions prior to 11.1.0) defining the number of groups in the file. If you do not specify *modulo*, `ICREATE.FILE` prompts for *modulo* and *separation*. *modulo* is ignored for nonhashed and dynamic files.
separation	An integer from 1 through 8,388,608 specifying the group buffer size in 512-byte blocks. If you do not specify *separation* and one is needed, the file is created with a separation of 4, unless you are in prompting mode. If you press Return in response to the *separation* prompt, the file is not created. *separation* is ignored for nonhashed and dynamic files.
description	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 *filename* by entering `?filename` . You must specify *description* in quotation marks when you create dynamic files.

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.
GROUP.SIZE *n*	Specifies the size of each group in the file, where *n* is the separation, which is a multiple of 2K of the value entered. The default is 1 for a separation of 2K.

Parameter	Description
LARGE.RECORD *n*	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%.
MERGE.LOAD *n*	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%.
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.
MINIMUM.MODULUS *n*	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.
RECORD.SIZE *n*	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.
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.
SPLIT.LOAD *n*	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%.

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.

`ICREATE.FILE` creates or modifies the file definition in the VOC file. In addition, `ICREATE.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:

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

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

```
ICREATE.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 `ICREATE.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 `ICREATE.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)	?.

Windows platforms

This character...	Maps to...
/	%S
?	%Q
empty filename	%
"	%D
%	%%

This character...	Maps to...
*	%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

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.

```
>ICREATE.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.

```
>ICREATE.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.

```
>ICREATE.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
ICREATE.FILE OVERDUE 2 1 1	OVERDUE 001 F 002 OVERDUE 003 D_OVERDUE
ICREATE.FILE DICT LONG.OVERDUE 4 3 2	LONG.OVERDUE 001 F 002 003 D_LONG.OVERDUE In PICK or REALITY flavor, the VOC entry appears as: LONG.OVERDUE 001 F 002 D_LONGOVERDUE 003 D_LONG.OVERDUE

Command	VOC entry
ICREATE.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

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	
expression	The test to be performed. Its syntax is *value1operatorvalue2*.	
	value1	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.
	operator	Any relational operator. For a complete list, see UniVerse keywords, on page 399.
	value2	Must be a constant, an @variable, or an inline prompt.
sentence	The sentence to execute if *expression* evaluates to TRUE.	

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.

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.

Syntax

INITIALIZE.CATALOG [-R]

Parameters

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.

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.

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.

Syntax

INITIALIZE.DEMO

JOBS

Use JOBS to list currently running phantom processes that were started from your current UniVerse process.

Phantom processes are processes that run in the background. You start them with the PHANTOM command.

There is a system-wide 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.

Syntax

JOBS

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.

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*.

Syntax

LIMIT [PROGRAMSIZE *size*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
size	Expressed as a number of 1024-byte units. Zero (0) specifies unlimited space.

LIST

Use LIST to display selected data from a file. You can sort and format the output of LIST in many ways.

LIST lists the selected records in the order in which they are stored in the file, unless you specify a sort expression.

Syntax

```
LIST [DICT | USING [DICT] dictname] filename [records | FROM n]
[selection] [output.limiter] [sort] [output] [report.qualifiers] [TOXML
[ELEMENTS] [WITHDTD] [XMLMAPPING mapping_file][TO xmlfile]] [TOJSON [TO
jsonfile]] [FORCE]
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description	
DICT	Lists records in the file dictionary of *filename*. If you do not specify DICT, records in the data file are listed.	
USING [DICT] *dictname*	If DICT is not specified, uses *dictname* as the dictionary of *filename*. If DICT is specified, the dictionary of *dictname* is used as the dictionary of *filename*.	
filename	The file whose records you want to list. You can specify *filename* 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.	
records	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 *n*	Specifies the records whose record IDs are stored in select list *n*.	
selection	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 WITH, on page 467.	
output.limiter	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 WHEN, on page 466.	
sort	A sort expression that specifies the type of sort and the field to sort on. The syntax is as follows:	
	BY *field*	Sort on *field* in ascending order.
	BY.DSND *field*	Sort on *field* in descending order.
	BY.EXP *field*	Explode multivalues in *field* and *sort* in ascending order.
For more information about sort expressions, see UniVerse keywords, on page 399. 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 *UniVerse NLS Guide*.		

Parameter	Description
output	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: AVG ENUM PCT BREAK.ON MAX TOTAL BREAK.SUP MIN TRANSPORT CALC You can follow a field name with one or more field qualifiers. Field qualifiers are: AS COL.HDG FMT ASSOC CONV MULTI.VALUE ASSOC.WITH DISPLAY.LIKE SINGLE.VALUE For information about these keywords and their synonyms, see UniVerse keywords, on page 399. If you do not specify *output*, fields specified in the @ phrase are listed. If there is no @ phrase in the file dictionary, only record IDs are listed.
report.qualifiers	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 UniVerse keywords, on page 399. If you do not specify any report qualifiers, `LIST` produces a report with: ▪ UniVerse default heading and footing ▪ Single spacing between records ▪ Column headings on every page ▪ No control breaks ▪ Paged output to the screen
TOXML	Outputs `LIST` results in XML format.
ELEMENTS	Outputs results in element-centric format.
WITHDTD	Output produces a DTD corresponding to the query.
XMLMAPPING *mapping_file*	Specifies a mapping file containing transformation rules for display. This file must exist in the &XML& file.
[TO *xmlfile*]	If you specify an *xmlfile* name, the `LIST` results are written to the file in the account's `&XML&` directory.

Parameter	Description
TOJSON [TO *jsonfile*]	Outputs `LIST` results in JSON format to the screen. If you specify a *jsonfile* name, the `LIST` results are written to the file in the account's `&XML&` directory.
FORCE	Forces the heading to display even if no records are selected.

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`*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
dist.filename	The name of a distributed file.

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*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
listname1	The name of a saved list in the &SAVEDLISTS& file. If you do not specify *listname1*, LIST.DIFF uses the list whose name is an empty string.

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.

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. 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 in the key store. 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.

If you enter LIST.FILE.STATS with no options, the command generates a listing of statistics stored in the UNIVERSE.STAT.FILE file.

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.

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] [INFORM | -INFORM | -INFORMME] [LPTR [n]] [NO.PAGE]
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file or table dictionary.
filename	The name of a UniVerse file or table. If you do not specify *filename*, LIST.INDEX prompts for it.
indexes	A list of indexed fields separated by spaces. *indexes* are the names of fields in *filename* used as secondary index keys.
ALL	Lists all the secondary indexes in the file.
STATISTICS | STATS | DETAIL	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. If you use DETAIL, UniVerse displays the name of each secondary key value, the number of records with that key value, and the bytes used for that key value.
INFORM | -INFORM | -INFORMME	Lists the physical path and physical index file names for each of the requested fields. It also indicates whether the index file is 32bit or 64bit, the size of the index, and the date on which it was last built. **Note:** When a data file is resized from 32bit to 64bit or vice-versa, the index file is not changed. An index needs to be deleted, recreated and rebuilt for it to match the 32bit or 64bit style of the data file in this situation. This option can be used with the ALL option. When using this option with the ALL option, output is sorted alphabetically by the dictionary field name not the INDEX.### name.
LPTR	Sends output to the printer via logical print channel *n*. If you do not specify *n*, 0 is assumed.
NO.PAGE	Suppresses automatic paging.

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: The information displayed from this command includes details from the dictionary definition, not what is in the built index. If the field's definition is different between the dictionary and what is built, the "In DICT" field in `LIST.INDEX` changes from Yes to No.

If the index is encrypted, UniVerse adds an "E" at the end of the "Type" field.

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	*fieldname*	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
	SQL	SQL index.
Build	Required	Index needs to be built
	Not Required	Index does not need to be built
Nulls	Yes	Empty strings are indexed
	No	Empty string are not indexed
In DICT	Yes	File dictionary contains corresponding field
	No	File dictionary does not contain corresponding field
S/M	S	Single-valued index
	M	Multivalued index
Just	L	Left or text justification
	R	Right justification
Unique	Y	Field or SQL index has UNIQUE column constraint
	N	Field or SQL index has no UNIQUE column constraint
Field num	*field number*	If Type is D, A, or S
I-type	*I-type expression*	If Type is I or SQL
	correlative	If Type is C
NLS Collation Locale	*locale.name*	The name of the locale associated with the index.

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*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
listname1	The name of a saved list in the &SAVEDLISTS& file. If you do not specify *listname1*, `LIST.INTER` 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.

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.

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.

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 *filename*. If you do not specify DICT, records in the data file are listed.

Parameter	Description	
USING [DICT] *dictname*	If DICT is not specified, uses *dictname* as the dictionary of *filename*. If DICT is specified, the dictionary of *dictname* is used as the dictionary of *filename*.	
filename	The file whose records you want to list. You can specify *filename* 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.	
records	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 *n*	Specifies the records whose record IDs are stored in select list *n*.	
selection	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 WITH, on page 467.	
sort	A sort expression that specifies the type of sort and the field to sort on. The syntax is as follows:	
	BY *field*	Sort on *field* in ascending order.
	BY.DSND *field*	Sort on *field* in descending order.
	BY.EXP *field*	Explode multivalues in *field* and *sort* in ascending order.
	BY.EXP.DSND *field*	Explode multivalues in *field* and *sort* in descending order.
	For more information about sort expressions, see UniVerse keywords, on page 399. 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 *UniVerse NLS Guide*.	
report.qualifiers	One or more of the following keywords: COL.HDR.SUPP HDR.SUP LPTR SAMPLE DBL.SPC HEADING MARGIN SAMPLED FIRST ID.SUP NOPAGE SUPP FOOTING These keywords modify the report format. For information about them and their synonyms, see UniVerse keywords, on page 399. If you do not specify any report qualifiers, `LIST.ITEM` produces a report with: ▪ UniVerse default heading ▪ Single spacing between records ▪ Paged output to the screen	

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 *filename*. If you do not specify DICT, records in the data file are listed.	
USING [DICT] *dictname*	If DICT is not specified, uses *dictname* as the dictionary of *filename*. If DICT is specified, the dictionary of *dictname* is used as the dictionary of *filename*.	
filename	The file whose records you want to list. You can specify *filename* 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.	
records	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 *n*	Specifies the records whose record IDs are stored in select list *n*.	
selection	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 WITH.	
output.limiter	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 WHEN. A sort expression that specifies the type of sort and the field to sort on. The syntax is as follows:	
	BY *field*	Sort on *field* in ascending order.
	BY.DSND *field*	Sort on *field* in descending order.
	BY.EXP *field*	Explode multivalues in *field* and *sort* in ascending order.
	BY.EXP.DSND *field*	Explode multivalues in *field* and *sort* in descending order.
	For more information about sort expressions, see 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 *UniVerse NLS Guide*.	
output	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 *output*, fields specified in the @ phrase are listed. If there is no @ phrase in the file dictionary, only record IDs are listed.	

Parameter	Description
report.qualifiers	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 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

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.
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.

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:

INDENT + COUNT(SIZE + 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 JANE LESLIE
        ADDAMS SAMUEL BROWN

ID 4108 5390 6100
NAME HILLARY ALICE BOB
            HENDERSON MIX MASTERS

ID 7100 4309 4439
NAME ALICE EDGAR DON
            WILLIAMS WILLIAMS ALISON

ID          5205        6203    6783
NAME        ALICE       SAM         DAVID
                CRATCHETT YORK      HALE

ID      7505
NAME    HARRY
            EDWARDS
13 records listed.
```

LIST.LOCALES

Use LIST.LOCALES in NLS mode to list information about available NLS locales.

LIST.LOCALES can use an active select list. If no select list is active, LIST.LOCALES lists all installed 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.
locale	Specifies a particular locale.

Parameter	Description
DETAIL	Lists the name and description of each locale followed by the locale definitions for each category.

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.

`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.

Syntax

`LIST.LOCKS`

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:-----
```

LIST.MAPS

Use `LIST.MAPS` in NLS mode to list information about available NLS maps.

Maps must be built and installed in shared memory. The listing includes the name and description of each map.

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.
mapname	Specifies a particular map.
DETAIL	Lists the map definitions in the NLS.MAPS.DESCS file.
BASE	Recursively lists any underlying base maps.

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 *terminal#*.
EVERY	Lists the active group locks in addition to the active file and record locks.

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 /etc/hosts 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. Beginning at UniVerse 11.2.3, LAN is no longer supported.
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 UVNet.
Login ID	The login name of the user who set the lock. The name is not returned for remote files accessed across UVNet.
Item-ID	The record ID of the locked record.

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.

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).
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.	

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

Example

The following example uses `LIST.READU EVERY`. When a user is holding a FILELOCK or READU lock while other users are attempting to acquire the same lock, `LIST.READU EVERY` displays the "Active File Waiters" and "Active Read Waiters" sections; otherwise, these sections do not display.

```
>LIST.READU EVERY

Active File Locks:
Device.... Inode....  Netnode Userno   Lmode          Pid Login Id
0063533057    103812        0     -4   14 FX    11403398 root

Active Group Locks:                                  Record Group Group Group
Device.... Inode....  Netnode Userno   Lmode G-Address.  Locks ...RD ...SH ...EX
0063533057    103815        0     -4   36 IN        400      1     0     0     0


Active Record Locks:
Device.... Inode....  Netnode Userno   Lmode        Pid Login Id Item-ID........
.....
0063533057    103815        0     -4   36 RU   11403398 root     TIME

Active File Waiters:                          Owner   Waiter
Device.............  Inode.............       Userno  Userno
 9223372200063533057              103812          -4      -6

Active Read Waiters:                          Owner   Waiter
Device.............  Inode.............       Userno  Userno
 9223372200063533057              103815          -4      -5
>
```

LIST.SICA

Use `LIST.SICA` to list the contents of the security and integrity constraints area (SICA) of an SQL table.

`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 UVNet.

Syntax

`LIST.SICA` *`tablename`* `[LPTR] [NO.PAGE]`

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
tablename	The name of an SQL table.
LPTR	Sends output to the printer.
NO.PAGE	Suppresses automatic paging.

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.

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

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 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. *Note*: If you specify a UID of 0 or -2, the LISTME output may contain users other than yourself.
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.

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.

The contents of the LISTU report depends on the operating system you are running.

Syntax

LISTU

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           ttyp0 Feb 17 16:54
rd              ttyp1 Feb 18 07:53
ellen           ttyp2 Feb 18 09:55
glen            ttyp3 Feb 18 08:03
tom             ttyp4 Feb 18 11:29
ted             ttyp5 Feb 18 08:11
qa              ttyp6 Feb 18 08:32
qa              ttyp7 Feb 17 17:19
wendy           ttyp8 Feb 18 08:37
steve           ttyp9 Feb 18 08:41
donna           ttypa Feb 18 08:42
sean            ttypb Feb 18 08:47
walter      ttypc Feb 18 12:28
katy            ttypd Feb 18 08:50
adrian      ttype Feb 18 08:52
qa              ttypf 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.
>
```

LISTUSER

Use LISTUSER to list all the users currently logged in with information such as user type and login timestamp.

Syntax

LISTUSER [-i | -n] [-a] [-t] [INTERNAL] [[*options*] *value*] [TERMINAL | PHANTOM | ME]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description	
-i | -n	Displays the IP address of the processes currently running. If you are adding a search for the IP address or SOAP port, or are searching for the TTY value, this option is automatically displayed.	
-a	Shows the account on a second line. This option is useful when a long account name is being used.	
-t	Displays TTY on a second line.	
options	Displays only specific search options, described below. Only one search option is allowed at a time, and each option must be followed by a *value*. For all search options, the full *value* is searched, for example `ACCT /usr/uv` would not show results in the `/usr/uv/HS.SALES` directory.	
	Search option	**Description**
	PORT | USER | USERNO	Displays the user number *value* in the UsrNo column.
	PID	Displays the process ID *value* in the Pid column. Pids are supported up to 9 characters long.
	UID	Displays the user ID *value* in the UID column.
	NAME | USERNAME	Displays the user name *value* in the UserName column.
	TYPE | USERTYPE	Displays the type in the UserType column. *value* can be TERMINAL or PHANTOM.
	TTY	Displays the terminal type *value* in the TTY column.
	IP | ADDR| IPADDR	Displays the IP address *value* in the IP-Address column.
	SOAP | SOAPPORT	Displays the SOAP port number *value* in the WsPort column.
	ACCT | ACCOUNT | WORKDIR	Displays the account name or directory *value* in the Acct column. If the account name or directory contains a space, enclose the *value* in double quotes. If necessary, use the -a option to display long account path names on a separate line.
TERMINAL | PHANTOM	Specify one of these options when using TYPE | USERTYPE to display the type of user.	
ME	Reports on only the current user when this command is used at TCL.	

Example

The following example displays `LISTUSER` with no options specified.

```
# LISTUSER
UsrNo Pid.... UID.. UserName Type Acct.............. LogonTime.............
```

```
    1   25161     0 root     Term /usr/uv              Mon Aug  1 12:31:41 2016
    2   25168     0 root     Term /usr/uv/HS.SALES     Mon Aug  1 12:32:30 2016

 Total lines returned: 2
```

The following example displays LISTUSER with a PID specified.

```
 # LISTUSER PID 25161
 UsrNo Pid.... UID.. UserName Type Acct................ LogonTime..............
    1   25161     0 root     Term /usr/uv              Mon Aug  1 12:31:41 2016

 Search lines returned: 1 (of 2)
```

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

LISTUN

LISTPH

LISTV

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.

Command	Description
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.

LOCK

Use LOCK to set any of the 64 task synchronization locks to the locked state.

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.

Syntax

LOCK *n* [PROMPT | NO.WAIT]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
n	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.

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.

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.

Syntax

LOG.RESTORE *first.log last.log* [*logdir.path*] [*device*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
first.log	The number of the first log file to restore.
last.log	The number of the last log file to restore.
logdir.path	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.
device	The device name of the tape drive. If you do not specify *device*, the device defined as MT0 in the &DEVICE& file is used.

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.

Use LOG.SAVE to save transaction logging files from disk to tape. LOG.SAVE does not change anything in the UV_LOGS file.

Syntax

LOG.SAVE *first.log last.log* [*logdir.path*] [*device*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
first.log	The number of the first log file to save.
last.log	The number of the last log file to save.
logdir.path	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.
device	The device name of the tape drive. If you do not specify *device*, the device defined as MT0 in the &DEVICE& file is used.

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

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
port	The name of the terminal on which to start up UniVerse. Prefix *port* with the string TTY (that is, TTY*port*) to make up the name of the port definition record, which must exist in the &DEVICE& file.
command	The command text. The command can be a stored sentence or paragraph or a UniVerse command. It cannot exceed 25 characters.

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#*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
-pid#	The process number of a phantom process to log out.

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*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
account	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.

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*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
account	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.

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.

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.

Syntax

```
LOOP
 statements
REPEAT
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
statements	UniVerse statements you want to execute repeatedly.

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
subject	The subject of your message. MAIL uses only the first word after the *-s* flag as the subject. Enclose subjects containing spaces in quotation marks.
user.list	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.

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] [*messagelist*]	Displays the specified messages.
n	Goes to next message and displays it.
e [*messagelist*]	Edits messages.
f [*messagelist*]	Lists heading lines of specified messages.
d [*messagelist*]	Deletes specified messages.
s [*messagelist*] *file*	Appends the specified messages to *file*.
u [*messagelist*]	Undeletes specified messages.
r [*messagelist*]	Replies to specified messages.
pre [*messagelist*]	Makes specified messages go back to /usr/mail.
m [*userlist*]	Sends mail to specific users.
q	Quits, saving unresolved messages in mbox.
x	Quits, does not remove system mailbox.
h	Lists active message headers.
?	Lists a brief summary of commands.
!	Shell escape.
c [*directory*]	Changes directory.

messagelist 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

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

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

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
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

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

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

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 a UniVerse Administrator logged in to the UV account to use MASTER.

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

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.

Parameter	Description
user.no	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.

MENU.DOC

Use MENU.DOC to produce a menu listing.

Syntax

MENU.DOC

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

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

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.

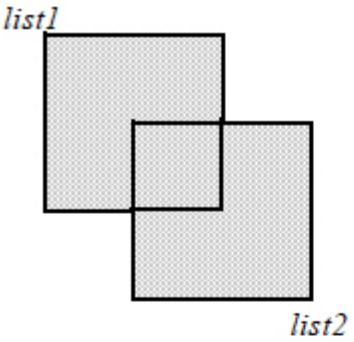
Parameter	Description
list1 list2	The numbers, 0 through 10, of the select lists you want to merge.
UNION	*list3* contains all elements from *list1* and all elements from *list2* that are not in *list1*.
INTERSECT [ION]	*list3* contains all elements from *list1* that are also in *list2*.
DIFF [ERENCE]	*list3* contains all elements from *list1* that are not in *list2*.
list3	The select list number for the output select list. If you do not specify *list3*, select list 0 is used.
COUNT.SUP	Suppresses the display of the number of records selected.

Use MERGE.LIST to create a select list from existing select lists. This avoids the need to execute another SELECT command.

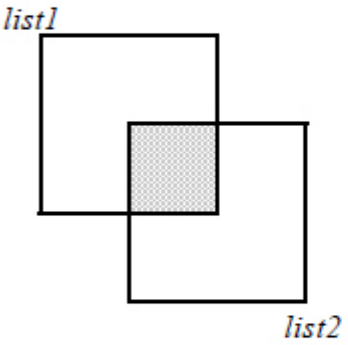
You must specify the numbers of the two select lists to be merged. If you specify a select list that is not active or is empty, MERGE.LIST issues a warning message and terminates.

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.

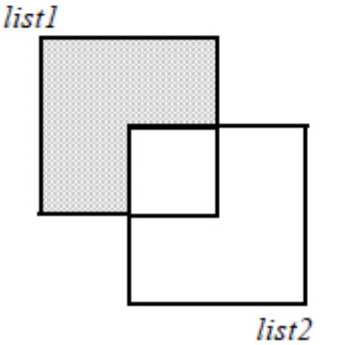
The relational set operations are illustrated in the following example. The shaded areas indicate what is included in the merged list.



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.

Example

1. Create the SELECT list.

   Create and save the select lists you want to merge. If you saved the lists to a list name, or if the lists already exist in the SAVEDLISTS file, use the GET.LIST command to retrieve the lists to a numbered list. In the following example, two lists are created and saved, then retrieved to a numbered list with the GET.LIST command:

   ```
   :SELECT CUSTOMER.F WITH CITY = "Denver"
   6 records selected to list 0.
   >SAVE.LIST DENVER
   6 key(s) saved to 1 record(s).
   :
   >SELECT CUSTOMER.F WITH CITY = "Boulder"
   2 record(s) selected to SELECT list #0.
   >>SAVE.LIST BOULDER
   2 record(s) SAVEd to SELECT list "BOULDER".
   >GET.LIST DENVER TO 1
   6 record(s) selected to SELECT list #1.
   >GET.LIST BOULDER TO 2
   2 record(s) selected to SELECT list #2.
   >
   ```

2. Merge the SELECT lists.

   Execute the MERGE.LIST command to merge the two active select lists. In the following example, MERGE.LIST creates an active select list of the INTERSECTION between list 1 and list 2.

   ```
   :MERGE.LIST 1 UNION 2
   3 record(s) selected.
   >SAVE.LIST DIFF
   ```

   In the previous example, no list number was specified to save the MERGE.LIST results, so MERGE.LIST stores the results to list 0. From the > prompt, you can execute the SAVE.LIST command to name an active select list to save.

   The next example illustrates how to specify a select list number to save the results of the MERGE.LIST command, then save the numbered list to a named list.

   ```
   :GET.LIST DENVER TO 1
   Overwriting existing select list.
   8 records retrieved to list 1.
   :GET.LIST BOULDER TO 2
   Overwriting existing select list.
   9 records retrieved to list 2.
   :MERGE.LIST 1 UNION 2 TO 3
   3 record(s) selected.
   :SAVE.LIST UNION FROM 3
   ```

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.

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.

Syntax

```
MESSAGE [user | number | -ACCEPT | -REJECT | -MSG text | -STATUS [ME] |
-SUPPRESS]
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
user	The user name of the person to whom you want to send a message. Windows Platforms. You can specify a full *domain**username* 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 *user*, MESSAGE sends the message to the system console.
number	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 *text*.
-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.

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
```

MKFILE.LIST

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.

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.

Syntax

MKFILELIST *account listname*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
account	The name of the account whose files you want to list. Define *account* in the UV.ACCOUNT file in the UV account.
listname	The record ID of the saved select list.

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*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
menu.name	The record ID of the VOC menu entry that invokes the menu. The options listed on the menu that you invoke must call submenus.

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`.

`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.

Syntax

NLS.ADMIN

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.

`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*.

Syntax

NLS.UPDATE.ACCOUNT

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 IGNORE is in effect.

Syntax

NOTIFY {ON | OFF} [IGNORE | INFORM]

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 (>).
IGNORE	After you issue the NOTIFY IGNORE command, PHANTOM processes exit completely when they have finished. UniVerse does not display a message that the PHANTOM completed.
INFORM	Displays the current NOTIFY setting.

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.

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*.

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.

Parameter	Description
filename	The name of the file against which NSELECT compares the currently active select list.
FROM *n*	Uses select list *n*. *n* is a number from 0 through 10. The default select list is 0.
TO *n*	Uses select list *n*. *n* is a number from 0 through 10. The default select list is 0.

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    1LIST 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.

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.

Syntax

OFF [*-pid#*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
-pid#	The process number of a phantom process to log out.

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.

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. The ON.ABORT entry will not be invoked if the abort occurs from a UniObjects (UCI) process.

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.

Syntax

ON.ABORT

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.

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. The `ON.EXIT` entry will not be invoked if the exit occurs from a UniObjects (UCI) process.

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.

Syntax

`ON.EXIT`

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
n	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`.

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.

When you log out, UniVerse automatically releases any devices, including printers, that have been assigned to you.

Syntax

P.DET *n*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
n	The number of the printer specified by a previous `P.ATT` command.

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.

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.

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.

PASSWD

Use `PASSWD` to set or change the password for your current login name.

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.
```

Syntax

PASSWD [*name*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
name	A login ID. If you do not specify *name*, your login ID is used by default. You must be a UniVerse Administrator to change a password you do not own.

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* -U *username* {*password*}

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.
command	The command text. The command can be a stored sentence or paragraph or a UniVerse command. *command* can be up to 25 characters.
-U *username* {*password*}	When you are logged in as root, specify -U *username* to run the phantom process as a different user. On Windows platforms, you must also specify the *password*.

There is a system-wide 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:
```

PHANTOM use that will consume a database license

The following UniVerse BASIC functions, which provide interactive capability when used in a PHANTOM, will consume a database license:

- UniVerse BASIC Socket API:
  - openSocket()
  - openSecureSocket()
  - initServerSocket()
  - initSecureServerSocket()
- UniVerse BASIC MQ API:
  - amInitialize()
- UniVerse BASIC CallHTTP
  - submitRequest()
- UniVerse BASIC CallSOAP:
  - SOAPSubmitRequest()
- TIMEOUT() when using microsecond timeout.

The license will be released when the PHANTOM exits. If no license is available, the process fails with a status code 99, indicating that UniVerse failed to obtain a license for the interactive PHANTOM process.

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 *name.* Windows Platforms. This is the user name, with or without the domain name; for example, `u2\jim` and `jim` are both acceptable.
PORT	Limits the report to jobs running on port *number*.
DEVICE	Limits the report to jobs attached to the device specified by *pathname*. Windows platforms do not support this option.
PID	Limits the report to the process specified by *process#*.
LOCK.HIST	Lists concurrency control operations.
ENABLE LOCK.HIST	Enables logging of concurrency control operations.
DISABLE LOCK.HIST	Disables logging of concurrency control operations.
LPTR	Sends the report to the printer.
RUNNING	Lists all UniVerse processes that are not yet suspended.
SUSPENDED	Lists all suspended UniVerse processes.

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.

Description

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.

Note: Starting with UniVerse v11.3.2, the `PORT.STATUS` command with `LAYER.STACK` option is available in the Python language layer support.

Refer to the *UniVerse Python Guide* for more information.

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 platforms: 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
File Group Record

Lock operation File No Group...    Lock Lock Lock Caller...
DBrsemv        4       0x00008400  R     D   DBread
DBrsemp        4       0x00008400  R     D   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/newsqlco [LOCALTEST @ 0x41C]
Data source.....................DBMS.........Node.............
rempubs                         UNIVERSE     pubs
```

The columns displayed are described in the following table.

Column	Description
Data source	Lists the name defined in the `uvodbc.config` file.
DBMS	Lists the type of database used to access the data.
Node	Lists the server the DBMS resides on.
Last command processed	The last TCL command run.
/dbms/enf/newsqlco [LOCALTEST @ 0x41C]	The current location in LOCALTEST.

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.

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.

Syntax

PRIME *n*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
n	The number you want to examine.

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.

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.

Syntax

PRINT.ADMIN

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. *channel* can be a number from 0 through 225. The default is 0.
MTU	Specifies the terminal characteristics of a magnetic tape channel. *channel* can be a number from 0 through 7. The default is 0.
DISPLAY	Displays a list of the current terminal characteristics.
option	The terminal characteristics you want to set.
setting	The new setting for the option.

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.
BGSTOP	Stops a background job if it tries to send output to the terminal.	
	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.
BREAK	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.
	IGNORE	Ignores the Break key, no interrupt is generated.
	NUL	The Break key inputs a NUL (ASCII 0) character.
BRK	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).
	OFF	Turns off the BRK character.
	c	Sets the BRK character to the character *c*.
BSDELAY	Sets the backspace delay.	
	ON	Specifies a delay of about .05 seconds whenever a backspace is output.
	OFF	Specifies no delay upon output of a backspace.

Option	Description	
CASE	Performs various conversions on uppercase and lowercase characters.	
	INVERT	Converts uppercase characters to lowercase and lowercase characters to uppercase on input.
	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.
	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.
	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.
	c	Sets the DSUSP character to the character *c*.

Option	Description	
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.
	OFF	Turns off Data Terminal Ready handshaking. Loss of DTR is treated as a hang-up, and foreground jobs are terminated.
ECHO	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.
	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.
EOF	Sets the end-of-file mark character. This character terminates input to many UNIX commands (*mail, dc*, 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.
	c	Sets the end-of-file mark character to the character *c*.
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.
	c	Sets the end-of-line mark character to the character *c*.
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.
	c	Sets the second end-of-line mark character to the character *c*.
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.
	c	Sets the character-erase character to the character *c.*

Option	Description	
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.
	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.
	c	Sets the flush character to the character *c*.
FMC	Sets the field mark character.	
	ON	Sets the field mark character to Ctrl-^.
	OFF	Turns off the field mark character.
	c	Sets the field mark character to the character *c*.
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.
	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.
	c	Sets the INTR character to the character *c*.

Option	Description	
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.
	c	Sets the KILL character to the character *c*.
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 and underscore (_) followed by a newline.	
	ON	Sets the line-continue character to Ctrl-_.
	OFF	Turns off the line-continue character.
	c	Sets the line-continue character to the character *c*.
LFDELAY	Sets the delay before a linefeed.	
	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 *opost 0*.
	OFF	Outputs characters without postprocessing. This setting is the same as *opost 1*.
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.
	c	Sets the literal-next character to the character *c*.
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.

Option	Description	
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	Treats 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.
	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.
	c	Sets the QUIT character to the character *c.*
RPRNT	Sets the reprint character. The reprint character redisplays 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.
	c	Sets the reprint character to the character *c*.
SMC	Sets the subvalue mark character	
	ON	Sets the subvalue mark character to Ctrl-\.
	OFF	Turns off the subvalue mark character.
	c	Sets the subvalue mark character to the character *c*.
SQLNULL	Sets the SQL null value character.	
	ON	Sets the null value character to Ctrl-N.
	OFF	Turns off the null value character.
	c	Sets the null value character to the character *c*.

Option	Description	
START	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.
	OFF	Turns off the START character.
	c	Sets the START character to the character *c*.
STOP	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.
	OFF	Turns off the STOP character.
	c	Sets the STOP character to the character *c*.
STOPBITS	Sets the number of stop bits on the terminal line protocol.	
	1	Sets the terminal line protocol for 1 stop bit.
	2	Sets the terminal line protocol for 2 stop bits.
STRIP	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.
	c	Sets the SUSP character to the character *c*.
SWTCH	Sets the SWTCH character. The SWTCH character is used in conjunction with *shl* to switch terminal input to the layering program (*shl*).	
	ON	Sets the SWTCH character to Ctrl-Z.
	OFF	Turns off the SWTCH character.
	c	Sets the SWTCH character to the character *c*.
TABS	Expands a tab character on output to the proper number of spaces. Tab stops are set every eight columns. Som 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.
	c	Sets the text mark character to the character *c.*
VMC	Sets the value mark character.	
	ON	Sets the value mark character to Ctrl-J.
	OFF	Turns off the value mark character.
	c	Sets the value mark character to the character *c*.

Option	Description	
VTDELAY	Sets the vertical tab delay.	
	ON	Specifies a two-second delay each time a vertical tab is output.
	OFF	Specifies no delay time when a vertical tab is output.
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.
	c	Sets the word-erase character to the character *c*.
XON	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.
	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.
	STARTANY	Any character acts as X-ON, if X-OFF/X-ON is enabled.
	NOSTARTANY	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.
	NOTANDEM	Turns off the automatic X-OFF/X-ON mode.

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 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
```

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. *channel* can be a number from 0 through 225. The default is 0.
MTU	Specifies the device characteristics of a device. name is the name of the device as specified in the &DEVICE& file.
DISPLAY	Displays a list of the current terminal characteristics.
option	The terminal characteristics you want to set.
setting	The new setting for the option.

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.	
	INVERT	Converts uppercase characters to lowercase and lowercase characters to uppercase on input.
	NOINVERT	No case conversions occur.
CRMODE	Sets the carriage-return mode.	
	INLCR	Converts carriage return to newline on input.
	NOINLCR	Prohibits conversion of carriage return to newline on input.

Option	Description	
ECHO	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.
	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.
	c	Sets the character-erase character to the character *c.*
FMC	Sets the field mark character.	
	ON	Sets the field mark character to Ctrl-^.
	OFF	Turns off the field mark character.
	c	Sets the field mark character to the character *c*.
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.
	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 NTR 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.
	c	Sets the INTR character to the character c. Not supported from a DOS command window.

Option	Description	
KILL	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.
	OFF	Turns off the KILL character. Not supported from a DOS command window.
	c	Sets the KILL character to the character *c*.
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 and underscore (_) followed by a newline.	
	ON	Sets the line-continue character to Ctrl-_.
	OFF	Turns off the line-continue character.
	c	Sets the line-continue character to the character *c*.
MODE	Determines how the input buffer handles input characters.	
	LINE	Input characters are not transmitted until a carriage return is received.
	RAW	Transmits input characters as they are received. This is the equivalent of setting INPUTMIN on and INPUTTIME to 0.
	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.
PARITY	Sets the parity.	
	NONE	Specifies that no parity generation is done for output, and enforces no parity checking on input.
	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 redisplays 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.
	c	Sets the reprint character to the character *c*.
SMC	Sets the subvalue mark character	
	ON	Sets the subvalue mark character to Ctrl-\.
	OFF	Turns off the subvalue mark character.
	c	Sets the subvalue mark character to the character *c*.

Option	Description	
SQLNULL	Sets the SQL null value character.	
	ON	Sets the null value character to Ctrl-N.
	OFF	Turns off the null value character.
	c	Sets the null value character to the character *c*.
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.
	c	Sets the text mark character to the character *c.*
VMC	Sets the value mark character.	
	ON	Sets the value mark character to Ctrl-J.
	OFF	Turns off the value mark character.
	c	Sets the value mark character to the character *c*.
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.
	c	Sets the word-erase character to the character *c*.
XON	Sets the X-ON and the X-OFF character.	
	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	Enables input X-OFF/X-ON processing.
	NOTANDEM	Disables input X-OFF/X-ON processing.

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
OL2 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 = 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
 >
```

PYTHON

Use PYTHON to launch into Python's interactive shell and execute Python commands.

Syntax

PYTHON

Examples

```
>PYTHON
python>
```

QSELECT

Use QSELECT to create a select list containing values from a specified field or from all fields of selected records.

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.

Syntax

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

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
filename	The name of a UniVerse file.
records	A list of record IDs.
*	Specifies all records in the file.

options can be any of the following:

Option	Description
SAVING *n*	Specifies a single field (the *n*th field) from which to extract values from each record.
TO *n*	Saves the select list in a numbered select list. *n* specifies the list number from 0 through 10.
FROM *n*	Supplies the *record.list* from a numbered select list. *n* specifies the list number from 0 through 10.

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.

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.

Syntax

Q

QUIT

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

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.

Addition codes

The following table describes the RADIX addition codes.

Code	Description
ADDX	Adds hexadecimal numbers.
ADDO	Adds octal numbers.
ADDB	Adds binary numbers.

Code	Description
ADDD	Adds decimal numbers.

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.

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.

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.

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
filename	The name of the file where the program is stored. RAID appends .O to *filename* to locate and operate on the object code. If you do not specify *filename*, RAID assumes the BP file by default.
program	The name of the record containing the source code of the program.

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.

Option	Description
TRAP	Enters the RAID debugger instead of displaying warning (nonfatal) error messages.

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
line	Displays the specified line of the source code.
/[*string*]	Searches the source code for *string.*
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.
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.
variable/	Prints the value of *variable.*
variable!*string*	Changes the value of *variable* to *string.*

RAID.GUI

Use RAID.GUI to set or change the way RAID displays output.

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.

Syntax

RAID.GUI [ON | OFF | SHOW]

RAID.GUI [START | STOP | END | INPUT] *string*

RAID.GUI [STARTHEX | STOPHEX | ENDHEX | 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.
STARTHEX	Specifies the start string in hexadecimal format.
STOPHEX	Specifies the stop string in hexadecimal format.
ENDHEX	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.

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
SHOWRAID 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.

`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.

Syntax

REBUILD.DF *dist.filename algorithm*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
dist.filename	The name of a distributed file.
algorithm	Can be either EXTERNAL, INTERNAL, or SYSTEM. See the DEFINE.DF command for details about the partitioning algorithm.

Use the following syntax to specify the partitioning algorithm:

Parameter	Description	
INTERNAL	The INTERNAL clause has the following syntax.	
	`INTERNAL{[[DATA] filename] record.ID| expression}`	
	DATA	Specifies that the algorithm expression is stored in the data file of *filename*. If you do not use DATA, the expression is stored in the dictionary of *filename*.
	filename	The name of the file containing the algorithm expression.
	record.ID	The ID of the record containing the algorithm expression. If you do not specify *filename*, the record is assumed to be an entry in the VOC file.
	expression	The I-type expression used as the partitioning algorithm. Enclose *expression*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 routine.name`	
	routine.name	The external routine used as the partitioning algorithm.
SYSTEM [*s*]	Specifies the default partitioning algorithm. *s* is the character used as a separator in record IDs. A hyphen (-) separator is the default.	

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.

 Examinging "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 3Compiling "@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.

If you omit a qualifier, RECORD prompts for it.

Syntax

RECORD [[DICT] *filename* [*record*]]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
filename	The name of the file.
record	The record you want to verify.

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 $UVHOME account to use this command.

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.

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

Parameter	Description
-F *file_path*	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 *file_path*. Do not use with the -M option.
-M *list_name* | -A	Specifies a *list_name* 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 *list_name*, or the -M parameter and -A. Do not use with -F option.
-C *starting_log_number ending_log_number*	Instructs the roll forward process to begin processing at *starting_log_number* and stop at the end of *ending_log_number*. During the recovery process, UniVerse verifies the log numbers. Use a space between the -C and *starting_log_number* and *ending_log_number*. Do not use with the -U option.
-U *starting_log_number ending_log_number*	Instructs the roll forward process to begin processing at *starting_log_number* and stop at the end of *ending_log_number*. During the recovery process, UniVerse does not verify the log numbers. Use a space between the -U and *starting_log_number* and *ending_log_number*. Do not use with the -C option.
-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 *record_ID*	Specifies a single record for UniVerse to recover. You must specify with the file name using the -F option. Use a space between the -I parameter and *record_ID*. If you do not specify a record ID, UniVerse displays an error and the program terminates. Do not use with the -N option.

Parameter	Description
-N *record_list_name*	Specifies a list of records stored in *record_list_name* 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 *record_list_name*. If you do not specify *record_list_name*, UniVerse displays an error and the program terminates. Do not use with the -I option.
-L *log_path*	Specifies the directory for log_path. If you do not specify the -L option, UniVerse uses the default log path.
-B *start_time*	Specifies the starting date and time within *starting_log_number* you specify with the -C or -U option from which to begin restoring the logs. Enter *start_time* in the yyyy-mm-ddThh:mm:ss or UTC format. The hour must be 00 - 23. Use a space between -B and *start_time*.
-E *end_time*	Specifies the ending date and time within ending_log_number you specify with the -C or -U parameter to stop restoring the logs. Enter *end_time* in the yyyy-mm-ddThh:mm:ss or UTC format. The hour must be 00 - 23. Use a space between -E and *end_time*.
-V *level*	Specifies the amount of messaging to output by the rollforward process. *level* 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 *level*, UniVerse displays an error and the process terminates.
-T *device*	Specifies the logs to recover on *device*. *device* must exist in the &DEVICE& file. To specify multiple tape devices, use a separate -T option followed by *device*. You must enter a space between -T and *device*. Do not use with the -L option.

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`.

`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.

Syntax

`RECOVERY.CHECKPOINT` {*`listname`* | `ALL` }

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
listname	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.

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`*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
pathname	The full pathname of a recoverable UniVerse file flagged as inconsistent. To determine which files are flagged as inconsistent, use the `RECOVERY.CHECKPOINT` command.

Example

This example clears the ORDERS file's inconsistency flag:

```
>RECOVERY.CONSISTENT /usr/accounts/ORDERS
```

REENCRYPT.FILE

Periodic file reencryption using different encryption keys is often needed due to internal policies or regulation. This process is often referred to as rekeying a file.

In order to rekey a file before this release, you had to first decrypt the file, then reencrypt it. At this release, decryption and encryption have been combined, reducing the time needed to rekey a file.

Use the `REENCRYPT.FILE` command to rekey a file.

Syntax

`REENCRYPT.FILE` *`<filename> <resize options>`* `-D`*`<enclist>`* `-E`*`<enclist>`*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
filename	The name of the file you want to reencrypt.
resize options	Any options available with the `RESIZE` command.
-D <*enclist*>	The list of fields you want to decrypt. Each field you specify in the *enclist* has the following format: <*field_name*> | WHOLERECORD,<*key*>[,*pass*>]
-W <*enclist*>	The list of fields you want to encrypt. Each field you specify in the *enclist* has the following format: <*field_name*> | WHOLERECORD,<*alg*>,<*key*>[,*pass*>]

The -D and -E options are independent of each other, meaning that you can specify different fields or modes can be different from the current definition. For example, you may want to change from WHOLERECORD encryption to field encryption.

If you specify nonencrypted fields in the -D encryption list, or the -E encryption list contains fields already encrypted but not decrypted by the -D list, UniVerse generates an error and terminates the process.

REENCRYPT.INDEX

The `REENCRYPT.INDEX` command encrypts and decrypts the index file associated with a field. This command only deals with the index file, it does not rebuild the index from the data file.

Syntax

`REENCRYPT.INDEX` *`<filename> -D <decrypt_specs> -E <encrypt_specs>`*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description	
filename	The name of the file for which you want to decrypt the index.	
descrypt_specs	*field*	The name of the field you want to decrypt.
	key	The actual key or file name containing the key;.
	pass	The password for the encryption key.
encrypt_specs	*field*	The name of the field you want to encrypt.
	alg	A string containing the cipher name.
	key	The actual key or file name containing the key.
	pass	The password for the encryption key.

You must specify the -D option before the -E option. Normally, you specify a field in both -D and -E to reencrypt the field. You can also only specify a field in -D or -E provided that:

- For -D option - The data field must be nonencrypted, and no @ID/WHOLERECORD encryption can exist in the file.
- For -E option - The index should already have been created for the field you specify, and it is not already encrypted.

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 *filename*. If you do not specify DICT, records in the data file are reformatted.
USING [DICT] *dictname*	If DICT is not specified, uses *dictname* as the dictionary of *filename*. If DICT is specified, the dictionary of *dictname* is used as the dictionary of *filename*.
filename	The file whose records you want to reformat. You can specify *filename* 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.
records	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 *n*	Specifies the records whose record IDs are stored in select list *n*.
selection	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 WITH, on page 467.
output.limiter	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 WHEN, on page 466.

Parameter	Description	
sort	A sort expression that specifies the type of sort and the field to sort on. The syntax is as follows:	
	BY *field*	Sort on *field* in ascending order.
	BY.DSND *field*	Sort on *field* in descending order.
	BY.EXP *field*	Explode multivalues in *field* and *sort* in ascending order.
	BY.EXP.DSND *field*	Explode multivalues in *field* and *sort* in descending order.
	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 UniVerse keywords, on page 399. 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 *UniVerse NLS Guide*.	
output	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.	
modifiers	One or more of the following keywords: FIRST ID.SUP SAMPLE ID.ONLY ONLY SAMPLED These keywords modify the report format. For information about them and their synonyms, see UniVerse keywords, on page 399.	
MTU *mtu*	Specifies a drive other than 0. *mtu* indicates the mode (*m*), number of tracks (*t*), and tape unit number (*u*) 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 the track specification is ignored. See MTU, on page 441.	
BLK *size*	Specifies the block size if it is different from the default of 8192 bytes.	
labelopt	One of the following: PICK.FORMAT (for Pick tape format) INFORMATION.FORMAT (for Prime INFORMATION tape format) REALITY.FORMAT (for REALITY tape format)	

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.

REINIT.PYTHON

The ECL REINIT.PYTHON command controls the implied re-initialization when invoking Python.

Note: Depending on the coding techniques and assumptions made in the Python code being used, the implied re-initialization of the Python environment might not be supported when used for a second time in the same session.

Syntax

REINIT.PYTHON [STATUS | ON | OFF]

Parameters

The following table describes each parameter of the syntax:

Parameter	Description
STATUS	Shows the current setting.
ON	(Default) Enables implied re-initialization when invoking Python.
OFF	Disables implied re-initialization when invoking Python.

Example

```
>REINIT.PYTHON STATUS
Current REINIT.PYTHON status: ON
```

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
filename	The name of the file from which to release the locks.
record	The record ID that you want to release the locks from.

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*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
logfile	The number of the log file you want to release for reuse. The log file's status must be Full.

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 system-wide 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.

REMOVE.DEMO.FILES removes the UniVerse data files, and the associated file dictionaries that were created by the MAKE.DEMO.FILES command.

Syntax

REMOVE.DEMO.FILES

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.

REMOVE.DEMO.TABLES removes the tables that were created by the MAKE.DEMO.TABLES command.

Syntax

REMOVE.DEMO.TABLES

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.
filename	The name of the file to be resized.
type	The new file type, a number (1 through 19, or 25) specifying one of the nondynamic file types.
modulo	An integer value greater than zero (limited to 8,388,608 in UniVerse versions prior to 11.1.0) defining the number of groups in the file. *modulo* is ignored if you specify a nonhashed or dynamic file type.
separation	An integer from 1 through 8,388,608 specifying the group buffer size in 512-byte blocks. If you do not specify *separation* and one is needed, the file is created with a separation of 4, unless you are in prompting mode. If you press Return in response to the *separation* prompt, the file is not created. *separation* is ignored for nonhashed and dynamic files.
CONCURRENT	Specifies that other users can use the file while it is being resized. This option is not supported on Windows platforms.
INPLACE	Resizes the file in its present physical location. This option does not exist when resizing to a type 30 file.
USING *partition*	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 /u4: `>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.

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.
GROUP.SIZE *n*	Specifies the size of each group in the file, where *n* is the separation, which is a multiple of 2K of the value entered. The default is 1 for a separation of 2K.
LARGE.RECORD *n*	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%.
MERGE.LOAD *n*	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%.

Parameter	Description
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.
MINIMUM.MODULUS *n*	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.
RECORD.SIZE *n*	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.
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.
SPLIT.LOAD *n*	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%.

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.

Note: If you resize a file from 32-bit to 64-bit and the file contains an index, the index will remain a 32-bit file. You must delete and re-recreate the index to change it to a 64-bit file.

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.

Note: Use of the RESIZE command on an audit log file will result in a different Inode on the resulting resized file than the Inode number of the originating file. This will cause U2 Audit Logging to stop. After you have resized the audit log file, you must use the -reloadonly option of the audman utility and make the RESIZEd file eligible for logging again. For example:
audman -config -reloadonly
Since the audit log file is a dynamic file, normally there is no need for you to manually resize the file. If you configure the audit log file to be sequential (or syslog on UNIX and Linux platforms), the hashed audit log file is not used. Therefore there is no need to resize it.

RESTORE.LOCALE

Use RESTORE.LOCALE in NLS mode to restore a previously saved locale setting.

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.

Syntax

RESTORE.LOCALE

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.

Parameter	Description	
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 processes:	
	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.
	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.	
verifield	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.	
	field	If you use a field name, ReVise verifies that the value entered in *verifield* is also a record ID in *verifile*.
	record.ID	If you use the record ID field name, ReVise verifies that the entered value is *not* a record ID in *verifile*. This does not work, however, if the ReVise sentence also includes a VERIFIELD expression. In such a case, *record.ID* must be a record ID in *verifile*.
verifile	The name of the verification file whose record IDs ReVise uses to validate the data entered in *verifield*.	
VERIFIELD	Displays the values in *verifile* that verify *verifield* data.	
display.field	The name of the field in *verifile* whose data ReVise displays for each value entered in *verifield*.	

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.ENCRYPTION.KEY

Use the `REVOKE.ENCRYPTION.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.ENCRYPTION.KEY` *`<key.id>`* `[`*`<password>`*`]` `{PUBLIC |`
`grantee` `{,`*`grantee...`*`}}`

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
key.id	The encryption key for which you are revoking access from another user.
password	The password for the encryption key for which you are revoking access from another user.
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.
grantee	Revokes access to the encryption key from the *grantee* you specify. *grantee* 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.

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
filename	The name of the file containing the source program. `RUN` appends .O to *filename*, then executes the specified program from this object file. If you do not specify *filename*, `RUN` assumes the BP file.
program	The name of the record containing the source code of the program. You must specify *program*.

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.
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.

Option	Description
TRAP	Enters the RAID debugger instead of displaying warning (nonfatal) error messages.

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.

RUNPY

Use RUNPY to execute a Python program.

Syntax

RUNPY [*filename*] *program* [*options*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
filename	The name of the file containing the source program. RUNPY appends .O to *filename*, then executes the specified program from this object file. If you do not specify *filename*, RUNPY assumes the BP file.
program	The name of the record containing the source code of the program. You must specify *program*.

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.
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 Python program.
TRAP	Enters the RAID debugger instead of displaying warning (nonfatal) error messages.

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.

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&, on page 478.

Syntax

SAVE.LIST [[*filename*] *listname*] [FROM *n*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
filename	The name of an existing type 1 or type 19 file. The default file is &SAVEDLISTS&.
listname	The name you are giving to the list. If you do not give the list a name, SAVE.LIST assigns the name &TEMP*port#*&. The *port#* is your terminal number.
FROM *n*	Specifies select list *n*. If you do not specify a FROM clause, select list 0 is saved.

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.

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*.

Syntax

SAVE.LOCALE

SAVE.STACK

Use SAVE.STACK to save the current sentence stack in the &SAVEDLISTS& file, under a specified name.

If you do not specify *listname*, SAVE.STACK prompts you for one.

Syntax

SAVE.STACK [*listname*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
listname	The name of the saved sentence stack.

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 filename. If you do not specify DICT, records in the data file are searched.
USING [DICT] *dictname*	If DICT is not specified, uses *dictname* as the dictionary of *filename*. If DICT is specified, the dictionary of *dictname* is used as the dictionary of *filename*.
filename	The file whose records you want searched. You can specify filename 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.
records	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 RetrieVe does not interpret them as field names or keywords.
FROM *n*	Specifies the records whose record IDs are stored in select list *n*.
selection	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 WITH, on page 467.

Parameter	Description	
output.limiter	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 WHEN, on page 466.	
sort	A sort expression that specifies the type of sort and the field to sort on. The syntax is as follows:	
	BY *field*	Sort on *field* in ascending order.
	BY.DSND *field*	Sort on *field* in descending order.
	BY.EXP *field*	Explode multivalues in *field* and *sort* in ascending order.
	For more information about sort expressions, UniVerse keywords, on page 399. 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 *UniVerse NLS Guide*.	
TO *n*	The number to assign to the list, 0 through 10. If you do not specify a number, SEARCH creates select list 0.	
options	One or more of the following:	
	ALL.MATCH	Selects only records containing all of the specified strings.
	NO.MATCH	Selects only records not containing any of the specified strings.
	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: *record.id*V*value*S*field*V0 The *record.id* is the record ID where the string is found, *value* and *field* 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.

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 *filename*. If you do not specify DICT, records in the data file are selected.
USING [DICT] *dictname*	If DICT is not specified, uses *dictname* as the dictionary of *filename*. If DICT is specified, the dictionary of *dictname* is used as the dictionary of *filename*.
filename	The file whose records you want to select. You can specify *filename* 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.
records	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 *n*	Specifies the records whose record IDs are stored in select list *n*.
selection	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 WITH, on page 467.
output.limiter	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 WHEN, on page 466.
sort	A sort expression that specifies the type of sort and the field on which field to sort. The syntax is as follows: BY *field* — Sort on *field* in ascending order. BY.DSND *field* — Sort on *field* in descending order. BY.EXP *field* — Explode multivalues in *field* and *sort* in ascending order. BY.EXP.DSND *field* — Explode multivalues in *field* and *sort* in descending order. For more information about sort expressions, see UniVerse keywords, on page 399. 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 *UniVerse NLS Guide*.
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.
field	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.

Parameter	Description
TO *n*	The number to assign to the list, 0 through 10. If you do not specify a number, SELECT creates select list 0.
report.qualifiers	One or more of the following keywords: FIRST LPTR SAMPLE SAMPLED These keywords modify the report format. For information about them, see UniVerse keywords, on page 399.

SEMAPHORE.STATUS

Use SEMAPHORE.STATUS to display the status of system semaphores. You can execute SEMAPHORE.STATUS only from the UniVerse account.

Syntax

SEMAPHORE.STATUS

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 /etc/hosts 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.

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.

If you do not specify qualifiers, `SET.FILE` prompts for them.

Syntax

SET.FILE [*account*] [*filename*] [*pointer*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
account	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.
filename	The name of a file in *account*.
pointer	The record ID of the Q-pointer in your VOC file.

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.
filename	The name or path of the file to which you want to assign a map.
mapname	The name of a map that is built and installed in shared memory. *mapname* 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 `uvconfig` 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.

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.LFILE 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 "ÀÁÂ"(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.FIPS.MODE

Use `SET.FIPS.MODE` to enable or disable FIPS mode while in a specific session.

If an invalid option is given with `SET.FIPS.MODE`, only the usage line is displayed.

@SYSTEM.RETURN.CODE returns 0 if `SET.FIPS.MODE ON` or `SET.FIPS.MODE OFF` succeeds, otherwise -1 is returned. If `SET.FIPS.MODE INFORM` or GET.FIPS.MODE is used, @SYSTEM.RETURN.CODE returns 1 if FIPS mode is on, or 0 if FIPS mode is off.

Syntax

`SET.FIPS.MODE` `{ON | OFF | INFORM} {BRIEF | HUSH}`

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
ON	Enables FIPS mode while in a specific session.
OFF	Disables FIPS mode while in a specific session.
INFORM	Reports the current FIPS mode. Using `SET.FIPS.MODE INFORM` returns the same results as the GET.FIPS.MODE command, which does not require any additional options and only reports the current status of FIPS mode.
BRIEF | HUSH	Suppress the output of `SET.FIPS.MODE`.

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

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*.

Syntax

SET.GCI.MAP [*mapname* | DEFAULT | NONE | OS]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
mapname	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 `uvconfig` 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 NLSDEFOSMAP configurable parameter in the `uvconfig` file.

SET.INDEX

Use `SET.INDEX` to change the characteristics of all the secondary indexes in a file.

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.

Syntax

SET.INDEX [DICT] *filename* [TO [*pathname* |NULL]] [TO RELATIVE.PATH] [*options*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Specifies the file dictionary.
filename	The name of a UniVerse file in which to change the characteristics of the secondary indexes. If a select list for *filename* is active, you do not need to specify *filename*.
pathname	The full path for the directory of the secondary indexes in *filename* to be changed in the file header.
NULL	Removes the current *pathname* for the indexes in *filename* from the file header.
TO RELATIVE.PATH	Changes the index location to be a path relative to the current directory. This is useful when you copy or move accounts from one location to another, as the path to the index file does not need to be changed.

options can be any of the following:

Option	Description
FORCE	Specifies that settings should be changed to *pathname* or NULL without prompting. If you do not specify FORCE, `SET.INDEX` verifies that you want to change or remove the path.
INFORM	Displays the current *pathname* for the indexes in *filename* in the file header on the terminal. You cannot use INFORM with another option.
OFF	Disables automatic secondary index updates for *filename*. Same as `DISABLE.INDEX`.
ON	Enables automatic secondary index updates for *filename*. Same as `ENABLE.INDEX`.

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 changes the path of the indexes for the INVENTORY file in the file header to be a relative path for the current directory and verifies that you want to make the change:

```
>SET.INDEX INVENTORY TO RELATIVE.PATH
The current indices for file "INVENTORY" are at path
/usr/uv/I_INVENTORY
to be changed to path
./I_INVENTORY
Do you wish to make this change (Y/N)? Y
File header block updated.
```

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/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.

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*.

Syntax

SET.LOCALE [*category* | ALL] {*locale* | OFF}

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
category	Sets the locale to one of the following categories: COLLATE CTYPE MONETARY NUMERIC TIME
ALL	Sets all categories. ALL is the default.
locale	The name of the locale you want to set.
OFF	Disables the locale for the designated category.

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.

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.

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.
devices	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 *devices*, CHECKPOINT mode is automatically set to OFF.

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
ONArchive 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
domain	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.
username	The user name to use for access to files on *nodename*. This must be a valid user name on the remote system. Windows Platforms. When you are connecting to a Windows system, *username* is not case-sensitive.
password	The password associated with *username* to use for access to files on *nodename*. *password* 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 *username*'s group membership on a remote UNIX system to *groupname*. *groupname* must be a valid group on the UNIX system, and username must be a member of the group.

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 *UVNet User 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
mapname	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 `uvconfig` file.
NONE	Uses internal format when reading or writing the file.

`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
1NF {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.

Option	Description	
ISOLATION *level*	Sets the isolation level for the current session. *level* 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 *bytes*	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.	
LOCK.WAIT *seconds*	Specifies the number of seconds to wait on a record or file lock before returning an error. The default values is 3600.	
NULL	UniVerse 11.2.5 and later. When the `SET.SQL` NULL option is followed by an empty set of quotes, the value is set to the default space value. For example, `SET.SQL NULL` "" sets the value to a single space. When the `SET.SQL` NULL option is followed by some character enclosed in quotation marks, the value is set to the specified value. For example, `SET.SQL NULL` "some character" sets the value to some character. When the `SET.SQL` command is not followed by an option, it displays the current setting for all options.	
OPTIM.SCAN {ON | OFF}	Turns optimistic scanning on or off. The default is ON.	
REPORTING	Lists the current SQL settings.	
SELECT.BUFFER *bytes*	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` 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.TELNET NODELAY

Use SET.TELNET NODELAY to set the socket option for a socket.

Syntax

SET.TELNET NODELAY [ON | OFF]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
ON	When SET.TELNET NODELAY is ON, the TCP_NODELAY socket option is set for the socket. This tells TCP/IP to always send the packet, regardless of its size. This may degrade performance on the physical network, but it will avoid the delay of waiting for an ACK.

Parameter	Description
OFF	When SET.TELNET NODELAY is OFF, the TCP_NODELAY socket option is not set, and TCP/IP waits to send the packet until it reaches a certain size, or a second packet is sent.

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	
code	The code for the type of terminal. It is case-sensitive. If you do not specify *code*, the current terminal type is assumed.	
options	One or more of the following:	
	AUTONL	Generates a new line sequence if a line longer than the width of the screen is typed.
	AUXMAP *mapname*	In NLS mode, sets a map for an auxiliary printer attached to the terminal. *mapname* is the name of the map you want to set. If you do not specify *mapname*, the map for the terminal is used.
	FUNDAMENTAL	Loads the default key bindings.
	HUSH	Suppresses terminal output.
	LENGTH *n*	Sets the length of the terminal screen to *n* lines.
	MAP *mapname*	In NLS mode, sets a map for the terminal. *mapname* is the name of the map you want to set.
	NEEDNL	Lets UniVerse generate new line sequences.
	VERIFY.SUP	Specifies no prompting is required if an invalid terminal type is entered.
	WIDTH *n*	Sets the width of the terminal screen to *n* characters. *n* must be an integer between 11 and 32,767.

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
pathname	The full path of the data file for which you are creating an entry.
filename	The name that you will use to access the file from this account.
OVERWRITING	Replaces an existing entry having the same name as *filename* 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.

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.

The following table describes each parameter of the syntax.

Parameter	Description
channel	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.
l.len	Set the line length, that is, the paper width. The default is 132.
p.len	Sets the number of lines per page. The default is 66.
top	Sets the top margin in number of lines. The default is 3.
bottom	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.

options can be any of the following:

Option	Description
AS [*name*]	Same as BANNER.
AT *name*	Routes output to *name* of a printer on a Windows system, using the format map name associated with the entry in the &DEVICE& file.
BANNER [*name*]	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 *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.

Option	Description
BANNER NEXT [*name*]	In mode 3, appends a sequential number to the name under which successive reports are created in &HOLD&. If you do not specify *name*, the record ID is P#0000_*nnnn*, where *nnnn* is increased for each new print job. If you specify *name*, the record ID is name_*nnnn*.
BANNER UNIQUE [*name*]	In mode 3, appends a sequential number to the name under which successive records are created in &HOLD&. If you do not specify *name*, the record ID is P#0000_*nnnn*, where *nnnn* is increased by each subsequent `SETPTR` command. If you specify *name*, the record ID is *name_nnnn*.
BRIEF	Suppresses the display of `SETPTR` settings.
COPIES *n*	Specifies the number of copies to print (with only one banner page).
DEFER *time*	Defers printing until *time*. Specify *time* in one of the following formats. The formats beginning with a plus sign (+) specify time relative to the current system time. 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
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 *name*	Uses the form *name* for printing this job. FORM prompts the user to put the named form in the printer.
FORMAT.MAP *mapname*	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.

Option	Description
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.
NOKEEP	Closes an open print file.
NORETAIN	In mode 1, turns off the RETAIN option.
PRINTER *name*	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 *options*	A string of one or more options to be sent to a printer driver script. *options* can be up to 128 characters long. If you specify more than one option, enclose *options* in double quotation marks.

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

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 *name* 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 `uvconfig` file.

Mode	Description
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 *mapname* for FORMAT.MAP, otherwise the format map is set from the default NLSDEFPTRMAP parameter in the `uvconfig` file.

For more information about character mapping, see the *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

The following table describes each parameter of the syntax.

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
channel	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.
l.len	Sets the line length, that is, the paper width. The default is 132.
p.len	Sets the number of lines per page. The default is 66.
top	Sets the top margin in number of lines. The default is 3.
bottom	Sets the bottom margin in number of lines. The default is 3.

Parameter	Description
mode	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.

For more information about character mapping, see the *UniVerse NLS Guide*.

options can be any of the following:

Option	Description
AS [*name*]	Same as BANNER.
AT *name*	Routes output to *name* of a printer on a Windows system, using the format map name associated with the entry in the &DEVICE& file.
BANNER [*name*]	In mode 1, a banner page is not produced unless the Windows printer has been configured to do so. If you do not specify name, the Windows printer banner is produced. If you specify *name*, 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 *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.
BANNER NEXT [*name*]	In mode 3, appends a sequential number to the name under which successive reports are created in &HOLD&. If you do not specify *name*, the record ID is P#0000_*nnnn*, where *nnnn* is increased for each new print job. If you specify *name*, the record ID is name_*nnnn*.
BANNER UNIQUE [*name*]	In mode 3, appends a sequential number to the name under which successive records are created in &HOLD&. If you do not specify *name*, the record ID is P#0000_*nnnn*, where *nnnn* is increased by each subsequent SETPTR command. If you specify *name*, the record ID is *name_nnnn*.
BRIEF	Suppresses the display of SETPTR settings.
COPIES *n*	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.

Option	Description
DEFER *time*	Defers printing until *time*. Specify *time* in one of the following formats. The formats beginning with a plus sign (+) specify time relative to the current system time. 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 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 *fontname*	Specifies the font to be used for printing in GDI mode. If you do not specify *fontname*, the printer default font is used.
FONTSIZE *points*	Used in conjunction with the *fontname*. Specifies the size of font (in points) for GDI mode. If *points* is not specified, a default size of 10 points is used.
FORM *name*	Uses the form *name* for printing this job. *name* 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 *mapname*	Sets a map name for formatting only. Data still goes to the spool queue with a map of NONE.
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 *lines*	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.

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>LNUM</td><td>Prints line numbers.</td></tr>
<tr><td>NFMT</td><td>Specifies that the application controls pagination and formatting instead of UniVerse.</td></tr>
<tr><td>NHEAD</td><td>Suppress printing a banner.</td></tr>
<tr><td>NODEFAULT</td><td>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 <code>SETPTR.DEFAULT</code> command.</td></tr>
<tr><td>NOEJECT</td><td>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.</td></tr>
<tr><td>NOFMT</td><td>Same as NFMT.</td></tr>
<tr><td>NOFORMFEED</td><td>Suppresses the addition of formfeeds to printer output.</td></tr>
<tr><td>NOHEAD</td><td>Same as NHEAD.</td></tr>
<tr><td>NOHOLD</td><td>In mode 1, turns off the HOLD option.</td></tr>
<tr><td>NOKEEP</td><td>Closes an open print file.</td></tr>
<tr><td>NORETAIN</td><td>In mode 1, turns off the RETAIN option.</td></tr>
<tr><td>PRINTER name</td><td>Same as AT.</td></tr>
<tr><td>PRIORITY</td><td>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>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>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>254</td><td>1</td></tr>
</table></td></tr>
<tr><td>RAW</td><td>Requests raw printing mode.</td></tr>
<tr><td>REQUEUE</td><td>Same as RETAIN.</td></tr>
</table>

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.

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 mapis 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 *name* 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 `uvconfig` 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 *mapname* for FORMAT.MAP. Otherwise, the format map is set from the default NLSDEFPTRMAP parameter in the uvconfig file.

Examples

```
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`.

`SETPTR.DEFAULT` uses the `SETPTR` settings of logical print channel 0 in the UV account to set system-wide default `SETPTR` parameters.

To specify system-wide 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 system-wide defaults to the initial system defaults.

Default `SETPTR` parameters are always reset to the initial system defaults after you reboot your system.

Syntax

SETPTR.DEFAULT

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.

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).

Syntax

SETUP.DEMO.SCHEMA *username*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
username	The user name whose account you want to convert into a schema. This user is the owner of the new schema.

SH

Use SH to invoke a UNIX Bourne shell (*sh*) from within UniVerse.

Note: This command is not supported on Windows platforms.

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.

Syntax

SH [-c "*command*" | *script*]

Parameters

The following table describes each parameter of the syntax.

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

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 06 11:43 I_CREDITORS
-rw-rw-r-- 1 susan acctg 1536 Jul 06 11:43 CREDITORS
-rw-rw-r-- 1 susan acctg 2048 Jul 06 11:42 D_CREDITORS
-rw-r--r-- 1 susan acctg 49152 Jul 06 11:41 VOC
drwxrwxrwx 2 susan acctg 512 Jul 02 16:55 I_ORDERS
-rw-r--r-- 1 susan acctg 3072 Jul 02 16:55 D_ORDERS
drwxrwxr-x 2 susan acctg 512 Jul 02 16:43
LONG.OVERDUE
-rw-rw-r-- 1 susan acctg 4096 Jul 02 16:41 D_LONG.OVERDUE
-rw-rw-r-- 1 susan acctg 2048 Jul 02 16:40 D_OVERDUE
-rw-rw-r-- 1 susan acctg 1536 Jul 02 16:40 OVERDUE
drwxrwxrwx 2 susan acctg 512 Jul 02 16:07 CUSTOMERS
-rw-r--r-- 1 susan acctg 3072 Jul 02 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 on to the UV account to use SHUTDOWN.RECOVERY.

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 reenable 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.

Syntax

SHUTDOWN.RECOVERY {INFORM}

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
INFORM	Displays messages during the shutdown process.

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.

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.

Syntax

`SLEEP` `[seconds | hh:mm[:ss]]`

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
seconds	Specifies the number of seconds to suspend processing. If you omit seconds, the default of 1 second is used.
hh:*mm*	Specifies the hour and minute when you want the process to restart. *hh* is in 24-hour format. You can also specify the seconds with :*ss*.

Example

This example suspends the current process until 3:30 p.m.:

```
>SLEEP 15:30
```

smat

Use `smat` as a diagnostic tool to examine information in the UniVerse shared memory structure. Use `smat` from an operating system command prompt.

Note: `smat` is the same command as the `analyze.shm` command. `smat` is shorthand for "Shared Memory Analysis Tool" and is used at the operating system level. For examples, see analyze.shm, on page 34.

Use the ANALYZE.SHM command in the UniVerse environment.

Syntax

smat {*options*}

Parameters

You must specify at least one option and precede each option with a hyphen. Options are case sensitive and must be preceded with a hyphen. For example -r shows record locks where -R shows U2 Data Replication status.

Parameter	Description
-a [*userno*]	(All) Lists information generated by all options. *userno* specifies a user number. You must be UniVerse Administrator to use the *userno* 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.
-n	(Numbers) Lists raw, unformatted data for all table entries, including unused entries.
-o	(Not supported for AIX) Lists only the OpenSSL version currently used by Universe.
-p [*userno*]	Display shared memory information for the specified user number. Requires UniVerse Administrator privileges.
-r	(Readu) Lists all active record locks. The UniVerse `LIST.READU` command also lists this information.
-R	(Replication) Lists information about the state of U2 Data Replication.

Parameter	Description

-s	(Semaphore) Lists the status of system semaphores. The `SEMAPHORE.STATUS` command also lists this information.
	
	Starting at 11.3.1, when U2 Audit Logging is enabled, -s shows audit logging daemon semaphores. The following example shows the semaphores for an audit of `uvaudd`:
	
	`ready 0`
	`mlock 1 owner 0`
	
	The following example shows the semaphores for an audit of buffer 0:
	
	`mutex 1 owner 0`
	`fillOK 1`
	`useOK 0`
	`active 0`
	
	The first column is the semaphore name; the second column is the current value of the semaphore.

-t [0]	(Tunables) Lists current configurable parameter values. 0 suppresses display of the asterisk that indicates the parameter has been changed from the default.
-u	(User) Lists the status of task synchronization locks. The UniVerse `LIST.LOCKS` command also lists this information.

-v	You can change the SYNCALOC and UVSYNC `uvconfig` parameters while UniVerse is running. Changes will take effect immediately without having to restart UniVerse. To do this, use the `analyze.shm` command with the -v option. Any parameter values that are changed will be displayed as shown in the following example:
	
	`analyze.shm -v "UVSYNC=0" "SYNCALOC=0"`
	`UVSYNC = 0`
	
	In the previous example, UVSYNC was changed from 1 to 0 so the new value was displayed. Because SYNCALOC was already set to 0, no change was made and a new value was not reported.
	
	Parameter changes made in this way are temporary and only apply to the currently running UniVerse run time and are discarded when UniVerse is stopped. To make the changes permanent, the `uvconfig` file must be updated.

Parameter	Description

-V	You can change the SYNCALOC and UVSYNC `uvconfig` parameters while UniVerse is running. Changes will take effect immediately without having to restart UniVerse. If you want to put the parameters to change in a file with the new values in a file, use the -V option. The following example illustrates a configuration file showing that the parameters and their associated values must be space delimited. Use of "=" as an assignment operator is not allowed in this context.
	
	`UVSYNC 1`
	`SYNCALOC 0`
	
	To apply the changes, use the following command:
	
	`bin/analyze.shm -V confparams`
	`SYNCALOC = 0`
	`UVSYNC = 0`
	
	This will generate output for each parameter that is changed, however if a parameter is not changed, then it will not be displayed, similar to the -v option.
	
	Parameter changes made in this way are temporary and only apply to the currently running UniVerse run time and are discarded when UniVerse is stopped. To make the changes permanent, the `uvconfig` file must be updated.

-w	Only lists processes waiting for file locks or record locks.
-x	Lists general system information not displayed by the other options.
-z	Includes network licensed users, when used with -c and -x.

SORT

Use `SORT` to display selected data from a file in sorted order.

`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.

Syntax

The following table describes each parameter of the syntax.

SORT [DICT | USING [DICT] *dictname*] *filename* [*records* | FROM *n*] [*selection*] [*output.limiter*] [*sort*] [*output*] [*report.qualifiers*] [TOXML [ELEMENTS] [WITHDTD] [XMLMAPPING *mapping_file*][TO *xmlfile*]] [TOJSON [TO *jsonfile*]] [FORCE]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
DICT	Lists records in the file dictionary of *filename*. If you do not specify DICT, records in the data file are listed.
USING [DICT] *dictname*	If DICT is not specified, uses *dictname* as the dictionary of *filename*. If DICT is specified, the dictionary of *dictname* is used as the dictionary of *filename*.

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>filename</td><td>The file whose records you want to list. You can specify filename 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.</td></tr>
<tr><td>records</td><td>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.</td></tr>
<tr><td>FROM n</td><td>Specifies the records whose record IDs are stored in select list n.</td></tr>
<tr><td>selection</td><td>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 WITH, on page 467.</td></tr>
<tr><td>output.limiter</td><td>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 WHEN, on page 466.</td></tr>
<tr><td>sort</td><td>A sort expression that specifies the type of sort and the field to sort on. The syntax is as follows:
<table>
<tr><td>BY field</td><td>Sort on field in ascending order.</td></tr>
<tr><td>BY.DSND field</td><td>Sort on field in descending order.</td></tr>
<tr><td>BY.EXP field</td><td>Explode multivalues in field and sort in ascending order.</td></tr>
</table>
For more information about sort expressions, see UniVerse keywords, on page 399.

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 UniVerse NLS Guide.</td></tr>
<tr><td>output</td><td>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:

AVG ENUM PCT

BREAK.ON MAX TOTAL

BREAK.SUP MIN TRANSPORT

CALC

You can follow a field name with one or more field qualifiers. Field qualifiers are:

AS COL.HDG FMT

ASSOC CONV MULTI.VALUE

ASSOC.WITH DISPLAY.LIKE SINGLE.VALUE

For information about these keywords and their synonyms, see UniVerse keywords, on page 399.

If you do not specify output, fields specified in the @ phrase are listed. If there is no @ phrase in the file dictionary, only record IDs are listed.</td></tr>
</table>

<table>
<tr><th>Parameter</th><th colspan="2">Description</th></tr>
<tr><td rowspan="13">report.qualifiers</td><td colspan="2">One or more of the following keywords:</td></tr>
<tr><td>COL.HDR.SUPP</td><td>FIRST</td></tr>
<tr><td>ID.ONLY</td><td>ONLY</td></tr>
<tr><td>COL.SPCS</td><td>FOOTING</td></tr>
<tr><td>ID.SUP</td><td>SAMPLE</td></tr>
<tr><td>COL.SUP</td><td>GRAND.TOTAL</td></tr>
<tr><td>LPTR</td><td>SAMPLED</td></tr>
<tr><td>COUNT.SUP</td><td>HDR.SUP</td></tr>
<tr><td>MARGIN</td><td>SUPP</td></tr>
<tr><td>DBL.SPC</td><td>HEADING</td></tr>
<tr><td>NO.SPLIT</td><td>VERT</td></tr>
<tr><td>DET.SUP</td><td>NOPAGE</td></tr>
<tr><td colspan="2">These keywords modify the report format. For information about them and their synonyms, see UniVerse keywords, on page 399.

If you do not specify any report qualifiers, SORT produces a report with:
▪ UniVerse default heading and footing
▪ Single spacing between records
▪ Column headings on every page
▪ No control breaks
▪ Paged output to the screen</td></tr>
<tr><td>TOXML</td><td colspan="2">Outputs SORT results in XML format.</td></tr>
<tr><td>ELEMENTS</td><td colspan="2">Outputs results in element-centric format.</td></tr>
<tr><td>WITHDTD</td><td colspan="2">Output produces a DTD corresponding to the query.</td></tr>
<tr><td>XMLMAPPING mapping_file</td><td colspan="2">Specifies a mapping file containing transformation rules for display. This file must exist in the &XML& file.</td></tr>
<tr><td>[TO xmlfile]</td><td colspan="2">If you specify an xmlfile name, the SORT results are written to the file in the account's &XML& directory.</td></tr>
<tr><td>TOJSON [TO jsonfile]</td><td colspan="2">Outputs SORT results in JSON format to the screen. If you specify a jsonfile name, the SORT results are written to the file in the account's &XML& directory.</td></tr>
<tr><td>FORCE</td><td colspan="2">Forces the heading to display even if no records are selected.</td></tr>
</table>

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.

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.

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 *filename*. If you do not specify DICT, records in the data file are listed.	
USING [DICT] *dictname*	If DICT is not specified, uses *dictname* as the dictionary of *filename*. If DICT is specified, the dictionary of *dictname* is used as the dictionary of *filename*.	
filename	The file whose records you want to list. You can specify *filename* 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.	
records	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 *n*	Specifies the records whose record IDs are stored in select list *n*.	
selection	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 WITH, on page 467.	
sort	A sort expression that specifies the type of sort and the field to sort on. The syntax is as follows:	
	BY *field*	Sort on *field* in ascending order.
	BY.DSND *field*	Sort on *field* in descending order.
	BY.EXP *field*	Explode multivalues in *field* and *sort* in ascending order.
	BY.EXP.DSND *field*	Explode multivalues in *field* and *sort* in descending order.
	For more information about sort expressions, see UniVerse keywords, on page 399. 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 *UniVerse NLS Guide*.	

Parameter	Description	
report.qualifiers	One or more of the following keywords:	
	COL.HDR.SUPP	HDR.SUP
	LPTR	SAMPLE
	DBL.SPC	HEADING
	MARGIN	SAMPLED
	FIRST	ID.SUP
	NOPAGE SUPP	FOOTING
	These keywords modify the report format. For information about them and their synonyms, see UniVerse keywords, on page 399. If you do not specify any report qualifiers, `SORT.ITEM` produces a report with: ▪ UniVerse default heading ▪ Single spacing between records ▪ Paged output to the screen	

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 *filename*. If you do not specify DICT, records in the data file are listed.
USING [DICT] *dictname*	If DICT is not specified, uses *dictname* as the dictionary of *filename*. If DICT is specified, the dictionary of *dictname* is used as the dictionary of *filename*.
filename	The file whose records you want to list. You can specify *filename* 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.
records	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 *n*	Specifies the records whose record IDs are stored in select list *n*.
selection	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 WITH, on page 467.
output.limiter	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 WHEN, on page 466.

Parameter	Description	
sort	A sort expression that specifies the type of sort and the field to sort on. The syntax is as follows:	
	BY *field*	Sort on *field* in ascending order.
	BY.DSND *field*	Sort on *field* in descending order.
	BY.EXP *field*	Explode multivalues in *field* and *sort* in ascending order.
	BY.EXP.DSND *field*	Explode multivalues in *field* and *sort* in descending order.
	For more information about sort expressions, see UniVerse keywords, on page 399. 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 *UniVerse NLS Guide*.	
output	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 *output*, fields specified in the @ phrase are listed. If there is no @ phrase in the file dictionary, only record IDs are listed.	
report.qualifiers	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 UniVerse keywords, on page 399. If you do not specify any report qualifiers, `SORT.LABEL` produces a report with: ▪ UniVerse default heading and footing ▪ Paged output to the screen	

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.

Specification	Description
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.

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:

```
INDENT + COUNT(SIZE + 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       EDGAR             DON
       HENDERSON         WILLIAMS          ALISON

ID  5205            5390               6100
NAME      ALICE           ALICE              BOB
       CRATCHETT         MIX             MASTERS

ID  6203           6783             7100
NAME      SAM           DAVID             ALICE
```

```
       YORK            HALE                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*] [R*unit*] [H] [S] [O] [T]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
formname	Sets the form name of *unit* to *formname*. You can use alphabetic and numeric characters to specify *formname*.
copies	The number of copies to print of each job.
?	Lists the current status. If you specify only ?, no unspecified options are changed to their defaults.
	As the last character of a UniVerse sentence, ? puts the sentence on the sentence stack without executing it. When you specify ? at the end of the command line, be sure to enter an extra space after the ? to prevent the sentence from being put on the sentence stack unexecuted.
D	Resets unspecified options to their default values.
F*form*	Sets the form number of *unit* to the number specified by *form*. Same as Q*form*.
Q*form*	Same as F*form*.
R*unit*	Indicates the logical print channel number.
H	Retains jobs in the spool queue after they are printed the first time.
	You can reprint held jobs using *usm* with the *-r* 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 *-k* option of *usm* 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 (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
copies	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.
F*form*	Sets the form number of *unit* to the number specified by *form*. Same as Q*form*.
Q*form*	Same as F*form*.
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.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*]] [F*form* [-*form*]] [L] [R] [U] [MD] [MS] [O]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
entry	The number of the entry to manipulate.
F*form*	Specifies only entries from the specified form numbers.
L	Lists the first 512 bytes of each spool file before the showing the Display prompt. This option overrides the action of the MD and MS options.
R	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.
U	Looks at files belonging to other users. You must be a UniVerse Administrator to use this option.
MD	Deletes all selected files.

Parameter	Description
MS	Spools all selected files.
O	Looks at active and waiting files and files in a held (or paused) state.

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 All.
Form	The form on which to print the job.
Status	The state of the job. This can be Active, Wait, or Hold. An asterisk (*) 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 (Yes or No).
Printer	The name of the printer. If no printer is specified for the job, this field contains [Any].
Size	The size of the file in bytes.
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.

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 `all` 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*]] [F*form* [*-form*]] [L] [R] [U] [MD] [MS] [O]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
entry	The number of the entry to manipulate.
F*form*	Specifies only entries from the specified form numbers.
L	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.
R	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.
U	Looks at files belonging to other users. You must be a UniVerse Administrator to use this option.
MD	Deletes all selected files.
MS	Spools all selected files.
O	Looks at active and waiting files as well as files in a held (or paused) state.

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 Paused or Printing.

Column Heading	Description
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.

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.

You must be sure that the tape unit is assigned to you (see ASSIGN, on page 46).

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.

Syntax

`SP.TAPE` [*printer*] [*alignment*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
printer	The name of a printer defined in the &DEVICE& file.
alignment	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 *alignment*, `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.

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.

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 filenamerecord -FORM TAPE
```

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 [ALL] *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
filename record	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. *printer* is case-sensitive. If you specify *printer*, only the jobs for that printer are listed; otherwise all jobs for all printers are listed.
-CANCEL	Deletes print jobs from the queue. *job#* is the number of the print job you want to cancel. You can specify multiple print job numbers, separated by spaces. Specifying –CANCEL ALL cancels all print jobs.
-AS	Prints *alias* instead of the filename on the flag page.
-AT	Specifies the name of the printer. *printer* 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. *form* can be up to thirty-two characters long.
-HEAD	Specifies to print the flag page.
-NOHEAD	Suppresses the flag page.

Parameter	Description
-PORTNO	Specifies the port number of the user who submitted the print job.
-PRINTER	Specifies the name of the printer. *printer* is case-sensitive.
-RANGE	Specifies a range of print jobs to cancel. *start* and *end* 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. *user* is case-sensitive.

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.

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
```

Syntax

SPOOL *filename record* [-AS *alias*] [-AT *printer*] [-COPIES *n*] [-FORM *form*] [-NOHEAD]

SPOOL -LIST [*printer*]

SPOOL -CANCEL [ALL] *job#* [*job#*] …

SPOOL -MODIFY *job#* [{HOLD | NOHOLD}] [PRIORITY *nnn*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
filename record	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. *printer* is case-sensitive. If you specify *printer*, only the jobs for that printer are listed; otherwise all jobs for all printers are listed.
-CANCEL	Deletes print jobs from the printer queue. *job#* is the number of the print job you want to cancel. You can specify multiple print job numbers separated by spaces. Specifying `-CANCEL ALL` cancels all print jobs.
-MODIFY	Changes the attributes of a print job already queued for printing. *job#* is the number of the print job you want to modify.
-AS	Prints *alias* instead of the filename on the flag page.
-AT	Specifies the name of the printer. *printer* 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. *form* 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.

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 *filename*. If you do not specify DICT, records in the data file are reformatted.
USING [DICT] *dictname*	If DICT is not specified, uses *dictname* as the dictionary of *filename*. If DICT is specified, the dictionary of *dictname* is used as the dictionary of *filename*.
filename	The file whose records you want to reformat. You can specify *filename* 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.

Parameter	Description	
records	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 *n*	Specifies the records whose record IDs are stored in select list *n*.	
selection	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 WITH, on page 467.	
output.limiter	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 WHEN, on page 466.	
sort	A sort expression that specifies the type of sort and the field to sort on. The syntax is as follows:	
	BY *field*	Sort on *field* in ascending order.
	BY.DSND *field*	Sort on *field* in descending order.
	BY.EXP *field*	Explode multivalues in *field* and *sort* in ascending order.
	For more information about sort expressions, see UniVerse keywords, on page 399. 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 *UniVerse NLS Guide*.	
output	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.	
modifiers	One or more of the following keywords: FIRST ID.SUP SAMPLE ID.ONLY ONLY SAMPLED These keywords modify the report format. For information about them and their synonyms, see UniVerse keywords, on page 399.	
MTU *mtu*	Specifies a drive other than 0. *mtu* indicates the mode (*m*), number of tracks (*t*), and tape unit number (*u*) 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 the track specification is ignored. See MTU, on page 441.	
BLK *size*	Specifies the block size if it is different from the default of 8192 bytes.	
labelopt	One of the following: PICK.FORMAT (for Pick tape format) INFORMATION.FORMAT (for Prime INFORMATION tape format) REALITY.FORMAT (for REALITY tape format)	

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 *filename*. If you do not specify DICT, records in the data file are selected.
USING [DICT] *dictname*	If DICT is not specified, uses *dictname* as the dictionary of *filename*. If DICT is specified, the dictionary of *dictname* is used as the dictionary of *filename*.
filename	The file whose records you want to select. You can specify *filename* 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.
records	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 *n*	Specifies the records whose record IDs are stored in select list *n*.
selection	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 WITH, on page 467.
output.limiter	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 WHEN, on page 466.

Parameter	Description	
sort	A sort expression that specifies the type of sort and the field to sort on. The syntax is as follows:	
	BY *field*	Sort on *field* in ascending order.
	BY.DSND *field*	Sort on *field* in descending order.
	BY.EXP *field*	Explode multivalues in *field* and *sort* in ascending order.
	BY.EXP.DSND *field*	Explode multivalues in *field* and *sort* in descending order.
	For more information about sort expressions, see UniVerse keywords, on page 399. 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 *UniVerse NLS Guide*.	
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.	
field	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 *n*	The number to assign to the list, 0 through 10. If you do not specify a number, `SSELECT` creates select list 0.	
report.qualifiers	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 UniVerse keywords, on page 399.	

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 *filename*. If you do not specify DICT, records in the data file are processed.
USING [DICT] *dictname*	If DICT is not specified, uses *dictname* as the dictionary of *filename*. If DICT is specified, the dictionary of *dictname* is used as the dictionary of *filename*.

Parameter	Description	
filename	The file whose records you want to process. You can specify *filename* 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.	
records	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 *n*	Specifies the records whose record IDs are stored in select list *n*.	
selection	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 WITH, on page 467.	
output.limiter	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 WHEN, on page 466.	
fields	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.	
report.qualifiers	One or more of the following keywords:	
	COL.HDR.SUPP	FIRST
	ID.SUP	SAMPLE
	COL.SPCS	FOOTING
	LPTR	SAMPLED
	COL.SUP	MARGIN
	SUPP	COUNT.SUP
	HEADING	ONLY
	HDR.SUP	NOPAGE
	VERT	DET.SUP
	ID.ONLY	
	These keywords modify the report format. For information about them and their synonyms, see UniVerse keywords, on page 399. If you do not specify any report qualifiers, STAT produces a report with: UniVerse default heading and footing Column headings on every page Paged output to the screen	

STATUS

Use `STATUS` to display information about users and resources.

Syntax

STATUS [*option*] [NO.PAGE]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
NO.PAGE	Suppresses automatic paging.

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.

If you do not specify any option, STATUS produces a full report (same as ALL).

stopuvsmm

Use stopuvsmm to stop the shared memory management tool.

Syntax

stopuvsmm [*options* | [-S | -F]] [0 | 1 | 2] [:*pid*]

Parameters

Most of the *options* available with stopuvsmm are reserved. The -r option is available to reauthorize the database or change the expiration date without stopping UniVerse. For example, to change the expiration date, use the following command:

uvregen -e *new_exp_date*

Then, while logged in as root (UNIX) or an administrator (Windows), enter stopuvsmm -r.

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 *filename*. If you do not specify DICT, records in the data file are processed.
USING [DICT] *dictname*	If DICT is not specified, uses *dictname* as the dictionary of *filename*. If DICT is specified, the dictionary of *dictname* is used as the dictionary of *filename*.
filename	The file whose records you want to total. You can specify *filename* anywhere in the sentence. SUM uses the first word in the sentence that has a file descriptor in the VOC file as the filename.
records	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 RetrieVe does not interpret them as field names or keywords.
FROM *n*	Specifies the records whose record IDs are stored in select list *n*.
selection	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 WITH, on page 467.
output.limiter	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 WHEN, on page 466.
fields	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.
report.qualifiers	One or more of the following keywords: 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 These keywords modify the report format. For information about them and their synonyms, see UniVerse keywords, on page 399. 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

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 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 on to the UV account to use SUSPEND.RECOVERY.

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.

Syntax

SUSPEND.RECOVERY {INFORM}

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
INFORM	Displays messages during the suspend process.

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.

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*.

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.
mtu	Indicates the mode (*m*), number of tracks (*t*), and tape unit number (*u*) of a tape drive. Only the unit specification is used, and the mode and track specifications are ignored. See MTU, on page 441.
size	Specifies the tape block size as *size*. 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, *size* 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.
mapname	Specifies the name of the map you want to set for the tape device. *mapname* must have been built and installed in shared memory. You can also specify *mapname* 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 *mapname* as the name in the NLSDEFDEVMAP parameter in the `uvconfig` file.

Parameter	Description
-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.BCK

Use `T.BCK` to back up a specified number of tape records or to back up over a previous end-of-file mark.

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.

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.
mtu	Indicates the mode (*m*), number of tracks (*t*), and tape unit number (*u*) of a tape drive. Only the unit specification is used, and the mode and track specifications are ignored. See MTU, on page 441.
nn	The number of tape records you want to back up over. If you do not specify *nn*, `T.BCK` backs up over the previous end-of-file mark.

T.DET

Use `T.DET` to relinquish exclusive control of the tape drive.

The `T.DET` command runs a BASIC program that executes an `UNASSIGN` command.

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.
mtu	Indicates the mode (*m*), number of tracks (*t*), and tape unit number (*u*) of a tape drive. Only the unit specification is used, and the mode and track specifications are ignored. See MTU, on page 441.

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.

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.

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.
filename	The name of the file containing the records you want to dump to tape. You can specify *filename* 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.
records	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 *n*	Specifies the records whose record IDs are stored in select list *n*.
selection	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 WITH, on page 467.

Parameter	Description	
sort	A sort expression that specifies the type of sort and the field to sort on. The syntax is as follows:	
	BY *field*	Sort on *field* in ascending order.
	BY.DSND *field*	Sort on *field* in descending order.
	BY.EXP *field*	Explode multivalues in *field* and *sort* in ascending order.
	BY.EXP.DSND *field*	Explode multivalues in *field* and *sort* in descending order.
	For more information about sort expressions, see UniVerse keywords, on page 399. 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 *UniVerse NLS Guide*.	
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.	
mtu	Indicates the mode (*m*), number of tracks (*t*), and tape unit number (*u*) of a tape drive. The default for these values, 000, indicates ASCII mode, 9-track tape, and unit number 0. Otherwise he track specification is ignored. See MTU, on page 441.	
BLK *size*	Indicates the block size if the block size is different from the default (exact size depends on the tape device you are using).	

modifiers can be any of the following:

Modifier	Description
HEADER *text*	Specifies *text* 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.
PICK.FORMAT	Specifies Pick tape format.
INFORMATION.FORMAT	Specifies Prime INFORMATION tape format.
REALITY.FORMAT	Specifies REALITY tape format.

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.

Parameter	Description
mtu	Indicates the mode (*m*), number of tracks (*t*), and tape unit number (*u*) of a tape drive. Only the unit specification is used, and the mode and track specifications are ignored. See MTU, on page 441.

T.FWD

Use `T.FWD` to advance a specified number of tape records or to the end-of-file mark.

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.

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.
mtu	Indicates the mode (*m*), number of tracks (*t*), and tape unit number (*u*) of a tape drive. Only the unit specification is used, and the mode and track specifications are ignored. See MTU, on page 441.
nn	The number of tape records to advance.

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.
filename	The name of the file into which you want to load the records. You can specify *filename* 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.
selection	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 WITH, on page 467.
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.

Parameter	Description
mtu	Indicates the mode (*m*), number of tracks (*t*), and tape unit number (*u*) of a tape drive. Only the unit specification is used, and the mode and track specifications are ignored. See MTU, on page 441.

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.

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.

`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.

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.
mtu	Indicates the mode (*m*), number of tracks (*t*), and tape unit number (*u*) of a tape drive. The default for these values, 000, indicates ASCII mode, 9-track tape, and unit number 0. The track specification is ignored. See MTU, on page 441.
n	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.READ

Use `T.READ` to read a tape record and then display it.

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.

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.
mtu	Indicates the mode (*m*), number of tracks (*t*), and tape unit number (*u*) of a tape drive. The default for these values, 000, indicates ASCII mode, 9-track tape, and unit number 0. The track specification is ignored. See MTU, on page 441.

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 *n*	Reads the tape beginning at tape record *n* relative to the current position of the tape.
TO *n*	Stops reading after tape record *n*.

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.
mtu	Indicates the mode (*m*), number of tracks (*t*), and tape unit number (*u*) of a tape drive. The default for these values, 000, indicates ASCII mode, 9-track tape, and unit number 0. The track specification is ignored. See MTU, on page 441.

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.
mtu	Indicates the mode (*m*), number of tracks (*t*), and tape unit number (*u*) of a tape drive. Only the unit specification is used, and the mode and track specifications are ignored. See MTU, on page 441.
n	The number of files to space forward on the tape. If you do not specify *n*, `T.SPACE` displays a prompt requesting the number of files.

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.

Parameter	Description
mtu	Indicates the mode (*m*), number of tracks (*t*), and tape unit number (*u*) of a tape drive. The default for these values, 000, indicates ASCII mode, 9-track tape, and unit number 0. The track specification is ignored. See MTU, on page 441.

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.
mtu	Indicates the mode (*m*), number of tracks (*t*), and tape unit number (*u*) of a tape drive. The default for these values, 000, indicates ASCII mode, 9-track tape, and unit number 0. The track specification is ignored. See MTU, on page 441.

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.	
mtu	Indicates the mode (*m*), number of tracks (*t*), and tape unit number (*u*) of a tape drive. The default for these values, 000, indicates ASCII mode, 9-track tape, and unit number 0. Otherwise he track specification is ignored. See MTU, on page 441.	
format	Specifies the tape label format. It can be one of the following:	
	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.

Parameter	Description
text	Specifies text to include in the tape label. *text* must be a single word. Do not enclose *text* in quotation marks. To include more than one word in text, separate the words with periods.

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 ("" represents an ASCII blank, I represents an item mark, and F represents a field mark):

I L "" xxxx "" HH:MM:SS

"" DD "" MMM "" YYYY

"" filename "" header pad F rr

PICK.FORMAT	
I	Item mark (segment mark)
L	ASCII character L
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 ("" represents an ASCII blank, I represents an item mark, F represents a field mark, and V represents a value mark):

I L header V HH:MM:SS

"" DD "" MMM "" YYYY

F rr F xxxxx F I pad

REALITY.FORMAT	
I	Item mark (segment mark)
L	ASCII character L
header	(Optional) Label header text (<= 44 characters)
V	Value mark (follows L if no header)
HH:MM:SS	Time (external 24-hour)
DD MMM YYYY	Date (external form)
F	Field mark (attribute mark)
rr	Reel number (decimal)
xxxxx	Tape record size (decimal)
pad	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.

Syntax

TANDEM *terminal.no*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
terminal.no	The terminal number of the terminal whose activity you want to watch or control.

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.

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.

Sequence	Description
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.

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.

If the target UniVerse session has TANDEM disabled (from the use of the DISABLE.TANDEM command, for example) and the TANDEM process tries to connect to that session, TANDEM reports the following message then exits:

```
Requested session has TANDEM operation disabled.
```

If the target UniVerse session interrupts the TANDEM connection by using the DISABLE.TANDEM FORCE command, the TANDEM process prints the following message and exits:

```
User has broken TANDEM connection. TANDEM terminated.
```

telnet

The UniVerse Unicode telnet command allows you to communicate with a remote computer that is using the Telnet protocol. This version of Telnet has the capability to input Chinese characters on Windows 7 and 8 platforms. Enter the telnet command to start the UniVerse Telnet Client.

Syntax

telnet [*RemoteServer*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
RemoteServer	Specifies the name of the server to which you want to connect.

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

The following table describes each parameter of the syntax.

#1 through *#8* are positional parameters. Each parameter is optional, but its position must be held by a comma.

Parameter	Description
#1	The line length of the terminal screen. *#1* must be an integer between 11 and 32,767. The default is a page width of 79.
#2	The number of lines displayed on the terminal screen. The default is a page depth of 24.
#3	The number of lines skipped at the bottom of the screen. This number is subtracted from page depth (*#2*). The default is 0.
#4	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.
#5	A number indicating the amount of delay following a formfeed. The default is 2. Range is from 2 through 3.
#6	The ASCII number corresponding to the Backspace key. The default is 8.
#7	The number of characters per line on the line printer. The default is 132.
#8	The number of lines per page on the line printer. The default is 66.
#9	The map name for the terminal device or the auxiliary printer.

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
icl6404
>
```

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

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*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
nnn	An octal number that specifies the file creation mask. If you do not specify a number, UMASK displays the current file creation mask.

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.

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.

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&, on page 474.

Syntax

```
UNASSIGN device
```

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
device	The device that you want to unassign. *device* must be defined in the &DEVICE& file.

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`.

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.

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 *node*	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 /etc/hosts file. Windows Platforms. If valid for Windows platforms, the path of the *hosts* file is `/Windows/System32/Drivers/etc/hosts` where Windows is the Windows installation directory. On Windows platforms, you can also use the LAN Manager node name. Beginning at UniVerse 11.2.3, LAN is no longer supported.
USER *user.number*	Restricts lock removal to locks owned by the specified user. You can get the user number from the output of LIST.READU.
FILE *pathname*	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 *device.number*	Restricts lock removal to a file or a device specified by the device number.
INODE *i.node.number*	Restricts lock removal to a file or a device specified by the i-node number.

Parameter	Description
GROUP *group.address*	Restricts lock removal to the group specified by the decimal *group.address*. The group address appears in hexadecimal format on the LIST.READU report and should be converted to decimal for use with this option.
RECORD *record.ID*	Restricts lock removal to the record specified by the record ID.

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 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`.

`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.

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
n	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 `uvconfig` 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.

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.

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 *filename*. If you do not specify DICT, converts records in the data file.
filename	The name of the file you want to convert. If *filename* is unmapped, it is converted to a mapped file with the specified *mapname*. If *filename* is mapped, it is converted to an unmapped file with a map name of NONE.
mapname	The name of the map the file uses for conversion from internal to external byte sequences. *mapname* must be built and installed in shared memory. Do not specify *mapname* 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 *mapname*. To use the default map file settings of NLSDEFFILEMAP and NLSDEFDIRMAP in the `uvconfig` file, specify *mapname* as DEFAULT. You cannot specify *mapname* 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.

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.
```

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 ]`

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.

The optional arguments PIOPEN, PICK, NEWACC, IN2, INFORMATION, and REALITY let you change the flavor of your account.

Note: You must exit and reenter your UniVerse session for the change of flavor to take effect.

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 in 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, if the first field of the VOC records do not match each other. 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.CER

The UPDATE.CER command updates the U2 default self-signed root certificate and its related security configuration. To use this command, you must be a UniVerse Administrator logged in to the UV account.

Syntax

UPDATE.CER

UniVerse has a default security configuration which is managed with the UPDATE.CER command. See the *Universe Security Features* guide for more information about this command.

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.
filenames	The names of UniVerse files whose indexes you want to update. If you do not specify *filenames*, UPDATE.INDEX prompts for one.

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 system-wide, 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] [-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 -p option, `usa` assumes all printers.
-F [a|d]	Mounts one or more forms on *printer*. *formlist* 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, -F replaces them with the new form list. -Fa adds the forms in *formlist* to the list of currently mounted forms. -Fd deletes the forms in *formlist* from the list of currently mounted forms. You must be a UniVerse Administrator or a member of a printer group to use -F.
-a	Used with the -F and -p options, -a verifies the alignment of *printer*. If the argument to -a is +*number*, `usa` uses the *number* of lines of the first waiting job of that form. If the argument is *pathname*, `usa` uses data in the file *pathname* for verification. *pathname* must be the file's absolute UNIX path. You must be a UniVerse Administrator or a member of a printer group to use -a. After printing an alignment test, `usa` 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 `err.log` and `act.log` files. You must be a UniVerse Administrator to use ±L.
-m *form*	Lists the status of all print jobs on printers using *form*.
-M	Allows users to merge the scattered usplogs at any time and then place the resulting merged file into the original path (`default:/usr/spool/uv`).

Parameter	Description
-n *port*	Lists the status of all print jobs on port. *port* 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.
-R	Simulates a crash restart by resetting the spooler daemon and rereading the `sp.config` and `usplog` files. After reading `usplog`, `usa` restores information about the spool queue to its precrash state. If changes were made to `sp.config`, `usa` 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 *user*	Lists the status of all print jobs of *user*. *user* 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.

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
directory	The name of the spooler directory. If you do not specify *directory*, the daemon uses `usr/spool/uv` as the spooler directory.
-a	Creates a new spooler activity logging file with the name *filename*. *filename* 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 *filename*. *filename* must be the full path of the error logging file.
-L	Creates default activity and error logging files `err.log` and `act.log` in the spooler directory (`/usr/spool/uv`).

Parameter	Description
-r	Specifies the number of seconds the spooler should wait after encountering an error condition before retrying the `usd` 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.

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 *string*. If *string* contains blanks, enclose the entire string in quotation marks.	
-BL	Begins printing at the line containing *string*. If *string* contains blanks, enclose the entire string in quotation marks.	
-E	Ends printing at the last character of *string*. If *string* contains blanks, enclose the entire string in quotation marks.	
-EL	Ends printing at the line containing *string*. If *string* contains blanks, enclose the entire string in quotation marks.	
-F	Changes the form to *formname*.	
-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 *copies*.	
-p	Moves a print job to the queue for printer *printer*.	
-P	Changes priority to *priority*.	
-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 *delay* when to print. You can specify delays in the following formats:	
	Syntax	**Example**
	+*minutes*	+5
	+*hours*:*minutes*	+2:15
	+*days*.*hours*:*minutes*	+7.00:00
	hour:*minute*	12:00
	day[*hour*:*minute*]	15.12:00
	month.*day*[.*hour*:*minute*]	6.15.12:00
	year.*month*.*day*[.*hour*:*minute*]	
-x	Prints page *n* or pages *n*-*m*.	
-y	Prints line *n* or lines *n*-*m*.	
jobnum	Specifies the print job number.	
all	Specifies all print jobs belonging to the user.	

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 *string*. If *string* contains blanks, enclose the entire string in quotation marks.	
-BL	Begins printing at the line containing *string*. If *string* contains blanks, enclose the entire string in quotation marks.	
-C	Prints *classtext* in banner line 3.	
-e	Suppresses the end-of-job page eject.	
-E	Ends printing at the last character of *string*. If *string* contains blanks, enclose the entire string in quotation marks.	
-EL	Ends printing at the last character of *string*. If *string* contains blanks, enclose the entire string in quotation marks.	
-f	Source file contains FORTAN forms control codes.	
-F	Waits until form *formname* is mounted on the printer.	
-h	Suppresses printing of the banner page.	
-H	Displays a short help message.	
-J	Prints *jobtext* in banner line 2.	
-l	Prints sequential line numbers.	
-n	Prints the number of copies specified by *copies*.	
-p	Routes to *printer* instead of to the default printer.	
-P	Sets *priority* 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:	
	+*minutes*	+5
	+*hours*:*minutes*	+2:15
	+*days*.*hours*:*minutes*	+7.00:00
	hour:*minute*	12:00
	day[.*hour*:*minute*]	15.12:00
	month.*day*[.*hour* : *minute*]	6.15.12:00
	year.*month*.*day*[.*hour* : *minute*]	95.6.15.12:00
-x	Prints page *n* or pages *n*-*m*.	
-y	Prints line *n* or lines *n*-*m*.	

Parameter	Description
files	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.

uv

Use the `uv` command 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. Starting at UniVerse 11.3.x on UNIX and Linux, the `uv` command is a script and will call `uvadmin` to perform any of the -admin functions.

Syntax

uv [-admin *option*] [-buildno] [-*version*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description	
-admin	You must be a UniVerse Administrator to use -admin and any of its options.	
option	One of the following:	
	-c[learshm]	Clears the BASIC catalog shared memory bit, allowing users to log in to UniVerse.
	-help	Lists all available -admin options.
	-start	Starts up UniVerse.
	-start_norpc	Starts up UniVerse with UniRPC off.
	-init	Starts the uvrepmanager with the -i option when used with the -start option.
	-stop	Shuts down UniVerse.
	-force [-now]	Forcefully shut down the UniVerse database. Use this option if UniVerse fails to shut down properly using only `uv -admin -stop`. During a normal shutdown, if user processes remain, the shutdown process exits and displays a message indicating that the shutdown did not complete. When you use the -force option, if any processes remain after a SIGTERM signal is sent, a SIGKILL signal is sent to the remaining processes after a short period of time. If you specify the -now option, the SIGKILL process signal is sent immediately to those processes that were not terminated by the SIGTERM signal.
	-nouser	Results in the shutdown process terminating with a message indicating the shutdown process did not take place if there were are any active user processes when the command was issued.
	-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.
	-D	Returns 0 if the database is active or 1 if the database is suspended.
-buildno	Returns the build number.	

Parameter	Description
-version	Displays the currently installed version of UniVerse.

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.

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.

`uvbackup` backs up files specified by *pathnames*. This means you must explicitly specify each data file and file dictionary you want to back up. Specify the paths of file dictionaries in the format D_*filename*. Index files are automatically backed up when you specify the *filename*.

Syntax

uvbackup {-d | -w | -f } [-notag] [-b *blksize*] [-cachedetail] [-cmdfil *filename*] [-noindex] [-delay *buffers*] [-l "*labeltext*"] [-limit buffers] [-rev7] [-rev8] [-rev93] [-rev94] [-rev95] [-rev102] [-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 `uvbackup`.
-w	(Weekly) Backs up records in specified UniVerse hashed files that have been added or changed since the last full or weekly `uvbackup`. 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.
-notag	Prevents the update of the two bits in the flag word of each record that determine the type of backup being performed. If a record indicates that it has not been backed up by `uvbackup`, this option leaves the bits the way they are, and the bits remain unchanged after the backup.
-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.
-noindex	Disables automatic backup of secondary indexes with files.

Option	Description
-delay	Specifies the number of buffers to use before flushing the buffer to the backup image. Generally, *buffers* is half the number of buffers specified by the -limit option. The default *buffers* value is 15.
-l	Specifies *labeltext* of up to 80 characters to identify the backup. You must enclose *labeltext* in quotation marks. All characters are valid. `uvbackup` 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 *buffers* 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.
-rev102	Produces a backup in a format suitable for restoring to UniVerse release 10.2.
-s	Specifies a file to which screen output is captured. You must also specify the -v or -V option to use -s.
-t	On UNIX, specifies the device to which to write backup data. *device* can be either a path or an entry in the &DEVICE& file. Use multiple -t options to specify up to 10 devices. `uvbackup` writes to the first device specified, then the second, and so on.
-v	Displays paths. `uvbackup` 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. `uvbackup` 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.
-	**UNIX only.** Reads paths from standard input.
pathnames	**UNIX only.** A list of files to back up. Each file name must be the path of the file.

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.

`uvbackup` backs up files specified by *pathnames*. This means you must explicitly specify each data file and file dictionary you want to back up. Specify the paths of file dictionaries in the format D_*filename*. Index files are automatically backed up when you specify the *filename*.

Syntax

uvbackup {-d | -w | -f } [-notag] [-b *blksize*] [-cachedetail] [-cmdfil *filename*] [-noindex] [-delay *buffers*] [-l "*labeltext*"] [-limit buffers] [-rev7] [-rev8] [-rev93] [-rev94] [-rev95] [-rev102] [-s *file*] [{-t *device*} ...] [-v | -V] [-walk *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 `uvbackup`.
-w	(Weekly) Backs up records in specified UniVerse hashed files that have been added or changed since the last full or weekly `uvbackup`. 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.
-notag	Prevents the update of the two bits in the flag word of each record that determine the type of backup being performed. If a record indicates that it has not been backed up by `uvbackup`, this option leaves the bits the way they are, and the bits remain unchanged after the backup.
-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.
-noindex	Disables automatic backup of secondary indexes with files.
-delay	Specifies the number of buffers to use before flushing the buffer to the backup image. Generally, *buffers* is half the number of buffers specified by the -limit option. The default *buffers* value is 15.
-l	Specifies *labeltext* of up to 80 characters to identify the backup. You must enclose *labeltext* in quotation marks. All characters are valid. `uvbackup` 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 *buffers* 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.
-rev102	Produces a backup in a format suitable for restoring to UniVerse release 10.2.
-s	Specifies a file to which screen output is captured. You must also specify the -v or -V option to use -s.
-t	On Windows, specifies that the output is directed to the named tape device. *device* can be either a path or an entry in the &DEVICE& file, for example, MT0. If you are backing up to disk, the file name you specify must already exist as an empty file.
-v	Displays paths. `uvbackup` 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. `uvbackup` 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 *pathnames*	**Windows only.** Generates a list of all the file paths to back up within the directories specified in the *pathnames* list, including all files in all subdirectories of the specified directories.

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"
```

uvcleanupd

The `uvcleanup` command forces a clean up of resources owned by a terminated process. You must be a UniVerse administrator or the user that has the same user number as the terminated process to execute the `uvcleanupd` command.

Syntax

uvcleanupd {[-t *time*] [-l *location*]} | {-c} | {-stop}

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
-t *time*	The interval time (in seconds) to have the `uvcleanupd` daemon check for dead processes. The default is 15 seconds.
-l *location*	The location of the uvcleanupd log file. The default location is `uvhome/uvcleanupd.log`.
-c	Instructs the `uvcleanupd` daemon to use the parameters that are defined in the `uvcleanupd.config` file on startup. When -c is specified, no other options can be used.
-stop	Stops the `uvcleanupd` daemon while UniVerse is running.

Note: If the `uvcleanupd` daemon is cleaning up a terminated `uvrw` process, the information logged in the `uvcleanupd.log` file will contain additional information regarding the replication log being processed. This can be helpful in diagnosing unexpected `uvrw` terminations.

uvdlockd

Use `uvdlockd` to administer the deadlock daemon. You must be a UniVerse Administrator to use `uvdlockd`.

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.

Syntax

uvdlockd {[-t *time*] [-r *res*] [-l *location*]} | [-query] | [-stop] | [-v *victim*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
-t *time*	The time interval (in seconds) between the deadlock daemon's successive checks of the lock-waiter tables. The default is 60 seconds.
-r *res*	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 *location*	The location of the deadlock log file. The default location is `uvhome/uvdlockd.log`.
-query	Generates a report based on a one-shot analysis of the lock-waiter tables and any detected deadlocks.
-v *victim*	Specifies which user number to select as the process to abort.
-stop	Shuts down the deadlock daemon.

Resolving deadlocks automatically

The deadlock daemon automatically resolves deadlocks by creating and updating a set of lock-waiter 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.

`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.

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 [level] *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 *setting*	Sets the case for interactive mode. *setting* 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 logfile instead of to standard error. If you do not specify logfile, `uvfixfile` creates a file called `uvfixlog` in the current directory.	
-m [ap]	Maps the primary buffer and overflow buffer layout.	
-o [output]	Records all output, including errors, to *outfile*. If you do not specify *outfile*, `uvfixfile` creates a file called *uvfixout* in the current directory.	
-rev	Displays the `uvfixfile` revision number.	
-s [tats]	Displays file statistics.	
-t [race]	Limits *groups* processed. *groups* can be on of the following:	
	n	Specifies one group.
	n-m	Specifies a range of groups.
	n+	Specifies a range of groups starting with *n* and ending at the end of the file.
	all	Specifies all groups.
	If you specify the -i options along with the -t option, -t is ignored.	
-v [level]	Sets the verbosity level of error messages. *level* can be a number from 0 through 9. See Verbosity levels for a list of levels.	

Parameter	Description
-z [ero]	Resets one or more group buffers. *groups* are the same as the options listed under the -t option.
-zgroup	Same as -z.

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

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
!*command*	Runs operating system *command.*
?	Displays available interactive mode commands with a brief description of each.
akpath [*pathname*]	Changes the path of the file's secondary indexes (alternate keys) to *pathname*. If you do not specify *pathname*, displays the current path of the file's secondary indexes. To set the secondary indexes' path to an empty string, specify *pathname* as NULL.
case *setting*	Sets input case. *setting* can be **invert** or **noinvert**.
ex [it]	Exits `uvfixfile` (same as quit).
file *pathname*	Opens a file specified by *pathname.*
free	Resets the freechain pointer.
gl	Sets group locks while tracing.
group.lock	Same as gl.
hash *record.ID*	Displays the group to which *record.ID* hashes.

Command	Description
help	Displays available interactive mode commands with a brief description of each.
locks	Displays if group locks will be set.
m [ap] *n*	Maps the 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 *pathname*	Opens a file specified by *pathname.*
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] *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. *groups* 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. *n* is a number from 0 through 9. If you do not specify *n*, `uvfixfile` displays the current verbosity level. See Verbosity Levels for a list of levels.
z [ero] *groups*	Resets one or more group buffers. *groups* 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.

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.
h [elp]	Displays available step commands.
q [uit]	Quits, and redisplays the interactive mode prompt (:).
set *parameter*	Sets various record block parameters. See Set command for a list of parameters you can set.
v [level] *n*	Sets the verbosity level to *n. n* is a number from 0 through 9. See Verbosity Levels for a list of levels.

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 [*value*]	Changes the backward link.
dec	Changes default base to decimal.
flagword [*value*]	Changes the flagword.
flink [*value*]	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.
t30lastb	Sets the dynamic file t30lastb bit.
transflg	(Obsolete) Sets transflg.
usenewpd	Sets the new style item padding flag.

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 *filename*.
filename	The name of a UniVerse file as defined in the VOC file. *filename* must be the first option on the command line. If a select list is active, you need not specify *filename*.
PATH	Specifies the relative of 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 *setting*	Sets the case for interactive mode. *setting* 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.

Parameter	Description	
HELP	Displays a list of available options.	
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 *logfile* 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 as synonym for REVISION.	
STATS	Displays file statistics. You can also use S to specify this option.	
TRACE	Limits *groups* processes. *groups* can be one of the following:	
	n	Specifies one group.
	n-m	Specifies a range of groups.
	n+	Specifies a range of groups starting with *n* 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 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

Level	Display
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

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
!*command*	Runs operating system *command.*
?	Displays available interactive mode commands with a brief description of each.
akpath [*pathname*]	Changes the path of the file's secondary indexes (alternate keys) to *pathname*. If you do not specify *pathname*, displays the current path of the file's secondary indexes. To set the secondary indexes' path to an empty string, specify *pathname* as NULL.
case *setting*	Sets input case. *setting* can be **invert** or **noinvert**.
ex [it]	Exits `uvfixfile` (same as quit).
file *pathname*	Opens a file specified by *pathname.*
free	Resets the freechain pointer.
gl	Sets group locks while tracing.
group.lock	Same as gl.
hash *record.ID*	Displays the group to which *record.ID* hashes.
help	Displays available interactive mode commands with a brief description of each.
locks	Displays if group locks will be set.
m [ap] *n*	Maps the 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 *pathname*	Opens a file specified by *pathname.*
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**).

Command	Description
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. *groups* 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. *n* is a number from 0 through 9. If you do not specify *n*, `uvfixfile` displays the current verbosity level. See Verbosity Levels for a list of levels.
z [ero] *groups*	Resets one or more group buffers. *groups* 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.

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.
h [elp]	Displays available step commands.
q [uit]	Quits, and redisplays the interactive mode prompt (:).
set *parameter*	Sets various record block parameters. See Set command for a list of parameters you can set.
v [level] *n*	Sets the verbosity level to *n*. *n* is a number from 0 through 9. See Verbosity Levels for a list of levels.

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 [*value*]	Changes the backward link.
dec	Changes default base to decimal.
flagword [*value*]	Changes the flagword.
flink [*value*]	Changes the forward link.
freeflag	Marks the record block as free.
grpresiz	Sets the relocated group bit.
hex	Changes default base to hexadecimal.

Parameter	Description
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.

uvgstt

The system-level `uvgstt` command displays the status and usage of global pages of shared memory.

See *Administering UniVerse* for more information on shared memory.

Note: This command is available starting at UniVerse 11.3.1.

Syntax

uvgstt

Examples

The following example illustrates a `uvgstt` command display:

```
# uvgstt
---------------------- GCTs Statistics --------------------
Total GCTs (GSMs allowed): 40
Pages/GSM.................: 32 (4096K bytes)
Bytes/Page................: 128K bytes
GCTs used (GSMs created).: 1 (3% of 40)

   Active GSMs....: 1 (32 pages in total, 4096K bytes)

        Pages Used...........: 2 (6%, 256K bytes)
        Pages Freed..........: 30 (94%, 3840K bytes)

   Inactive GSMs..: 0

        Pages Freed..........: 0 (0K bytes)

   Total Pages Used......: 2 (6%, 256K bytes)
   Total Pages Freed.....: 30 (94%, 3840K bytes)
   Total memory allocated: 4096K bytes
----------------- End of GCTs Statistics ----------------
```

uvipcrm

The system-level `uvipcrm` command removes all interprocess communication (IPC) structures associated with UniVerse. Execute this command at the system prompt.

If you are running multiple versions of UniVerse, `uvipcrm` removes only the structures associated with the version from which you execute `uvipcrm`.

Note: This command is supported on UniVerse for UNIX only.

Warning: Running `uvipcrm` stops all UniVerse background processes and halts all UniVerse user processes.

Syntax

uvipcrm

uvipcstat

Windows only. The system-level `uvipcstat` command displays the status of interprocess communication (IPC) facilities. In addition, UniVerse provides the names of the UniVerse processes associated with each resource.

For detailed information about this utility, see the section on managing IPC facilities in *Administering UniVerse*.

Note: Use this command at the system prompt.

Syntax

uvipcstat [-q] [-m] [-s] [-g] [-b] [-c] [-o] [-p] [-t] [-a] [-n]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
no parameter	Displays the status of all message queue, shared memory, and semaphores.
-q	Displays the status of message queues.
-m	Displays the status of shared memory.
-s	Displays the status of semaphores.
-g	Displays the UniVerse signals. This parameter is only supported on Windows platforms.
-b	Displays the status of the largest size allowed in each setting: the number of bytes on message queues, the size of the segments in shared memory, and the number of processes attached to each memory segment. This parameter is only supported on UNIX platforms.
-c	Displays the creator's login name and group name. This parameter is only supported on UNIX platforms.
-o	Displays the usage information of each of the following: the number of bytes on message queues, and the number of semaphores in each set. This parameter is only supported on UNIX platforms.

Parameter	Description
-p	Displays information about a process ID number: the process ID of the last process to send or receive messages on the message queue, the process ID of the creating process, and the final process to attach or detach on shared memory segments. This parameter is only supported on UNIX platforms.
-t	Displays the time information about: the time of the last control operation which changed access permissions for all facilities, the time of the final msgsnd and msgrcv on the message queues, the time of the ending shmat and shmdt on shared memory, and the time of the final semop on the semaphore sets This parameter is only supported on UNIX platforms.
-a	Displays the -b, -c, -o, -p and -t options.
-n	Displays the message queues with more than zero bytes in any currently outstanding messages. If a queue has zero bytes, it is not listed. This parameter only works with the -a or -o options. This parameter is only supported on UNIX platforms.

Note: Use `uvipcstat -qon` on UNIX to only display message queues with any bytes in the queue at the moment the command was run. On large UniVerse installations (those with a large number of users), you can end up with hundreds of message queues to page through in an `uvipcstat -qa` listing.

Examples

The following example shows an `uvipcstat` display:

```
# bin/uvipcstat
IPC status from /dev/mem as of Wed Jul 20 23:13:55 MDT 2016
T         ID     KEY        MODE       OWNER    GROUP
Message Queues:
q          0 0x4107001c -Rrw-rw----     root   printq  -> unknown
q   13631489 0xffffffff -Rrw-rw-rw-     root   system  -> uvrm R11.3 (request)
q   15728642 0xace0ad11 -Rrw-------     root   system  -> unknown
q   15728643 0xffffffff -Rrw-rw-rw-     root   system  -> uvsmm R11.3 (request)
q   12582916 0xffffffff --rw-rw-rw-     root   system  -> uvrm R11.3 (reply)
q   15728645 0xffffffff --rw-rw-rw-     root   system  -> uvsmm R11.3 (reply)
q   12582918 0xacea0207 -Rrw-rw-rw-     root  7602412  -> uv R11.3 (spooler)

Shared Memory:
m          0 0xffffffff D-rw------- pconsole   system  -> unknown
m   27262977 0xffffffff --rw-rw-rw-     root   system  -> uvsmm R11.3 (shmbuf)
m          3 0x61003007 --rw------- pconsole   system  -> unknown
m   17825804 0x780007ad --rw-rw-rw-     root   system  -> unknown
m   13631502 0x78000100 --rw-rw-rw-     root   system  -> unknown
m    2097171 0x7800020a --rw-rw-rw-     root   system  -> unknown
m    2097176 0x78100100 --rw-rw-rw-     root   system  -> unknown
m   36700192 0xffffffff --rw-rw-rw-     root   system  -> uvrm R11.3 (ctl)
m  180355116 0xacecb300 --rw-rw-rw-     root   system  -> uv R11.3 (disk)
m   20971585 0x0d000c13 --rw-rw----     root   system  -> unknown
m   11534515 0xace0ad01 --rw-rw-rw-     root   system  -> unknown
Semaphores:
s    3145728 0x0104002a --ra-ra-r--     root   system  -> unknown
s          1 0x620000ed --ra-r--r--     root   system  -> unknown
s          2 0xffffffff --ra-------     root   system  -> unknown
s          3 0x02000779 --ra-ra-ra-     root   system  -> unknown
s          4 0x01000779 --ra-ra-ra-     root   system  -> unknown
```

```
s  16777222 0xffffffff --ra-ra-ra-     root   system  -> uvrm R11.3 (waiting)
s  16777223 0xffffffff --ra-ra-ra-     root   system  -> uvrm R11.3 (waiting)
s  16777224 0xffffffff --ra-ra-ra-     root   system  -> uvrm R11.3 (waiting)
s  16777225 0xffffffff --ra-ra-ra-     root   system  -> uvrm R11.3 (waiting)
s        15 0x81003006 --ra------- pconsole   system  -> unknown
s   7340048 0xffffffff --ra-ra-ra-     root   system  -> unknown
s  13631505 0xffffffff --ra-ra-ra-     root   system  -> uvrm R11.3 (waiting)
s  13631506 0xffffffff --ra-ra-ra-     root   system  -> uvrm R11.3 (waiting)
s  13631507 0xffffffff --ra-ra-ra-     root   system  -> uvrm R11.3 (waiting)
s  13631508 0xffffffff --ra-ra-ra-     root   system  -> uvrm R11.3 (waiting)
s  13631509 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (ctl)
s  15728662 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (uvcleanupd
)
s  10485783 0xffffffff --ra-ra-ra-     root   system  -> unknown
s  15728664 0xace0adb1 --ra-ra-ra-     root   system  -> unknown
s  13631513 0xace0adb2 --ra-ra-ra-     root   system  -> unknown
s  13631514 0xace0adb3 --ra-ra-ra-     root   system  -> unknown
s  11534363 0xffffffff --ra-ra-ra-     root   system  -> unknown
s  13631516 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (latch)
s  13631517 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (latch)
s  13631518 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (latch)
s  13631519 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (latch)
s  13631520 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (latch)
s  13631521 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (latch)
s  13631522 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (latch)
s  13631523 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (latch)
s  13631524 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (latch)
s  13631525 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (latch)
s  13631526 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (latch)
s  13631527 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (latch)
s  13631528 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (latch)
s  13631529 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (latch)
s   5242922 0xffffffff --ra-ra-ra-     root   system  -> unknown
s  13631531 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (latch)
s  13631532 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (latch)
s  13631533 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (latch)
s  13631534 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (latch)
s  13631535 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (latch)
s  13631536 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (latch)
s        49 0xffffffff --ra-ra-ra-     root   system  -> unknown
s  14680114 0xffffffff --ra-ra-ra-     root   system  -> uvrm R11.3 (waiting)
s   1048627 0xffffffff --ra-ra-ra-     root   system  -> unknown
s        52 0xa10001a1 --ra-ra-ra-     root   system  -> unknown
s  11534393 0xace0adb6 --ra-ra-ra-     root   system  -> unknown
s  10485818 0xffffffff --ra-ra-ra-     root   system  -> uvrm R11.3 (waiting)
s  14680125 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (latch)
s   4194366 0xffffffff --ra-ra-ra-     root   system  -> unknown
s  14680127 0xffffffff --ra-ra-ra-     root   system  -> uvrm R11.3 (waiting)
s  14680128 0xffffffff --ra-ra-ra-     root   system  -> uvrm R11.3 (waiting)
s  14680129 0xffffffff --ra-ra-ra-     root   system  -> uvrm R11.3 (waiting)
s  15728706 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (latch)
s  14680131 0xffffffff --ra-ra-ra-     root   system  -> uvrm R11.3 (waiting)
s  13631556 0xffffffff --ra-ra-ra-     root   system  -> uvrm R11.3 (waiting)
s  13631557 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (latch)
s   9437254 0xffffffff --ra-ra-ra-     root   system  -> unknown
s  14680135 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (latch)
s  14680136 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (latch)
s  14680137 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (latch)
s  14680138 0xffffffff --ra-ra-ra-     root   system  -> uvsmm R11.3 (latch)
s  14680139 0xace0adb4 --ra-ra-ra-     root   system  -> unknown
s  13631564 0xffffffff --ra-ra-ra-     root   system  -> uvrm R11.3 (waiting)
s  13631565 0xffffffff --ra-ra-ra-     root   system  -> uvrm R11.3 (waiting)
```

```
s  13631566 0xffffffff --ra-ra-ra-     root   system  -> uvrm R11.3 (waiting)
```

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`.

Note: Starting at UniVerse 11.3.1, the IPv6 protocol is supported. IPv6 compatibility provides the ability to create more IP addresses, as IPv4 addresses are running out. In addition, IPv6 utilizes a more secure data package. However, UniVerse device licensing is not supported on IPv6 protocols at this time. If you are connected using an IPv6-only client, the `uvlictool` command displays 127.0.0.1 (localhost) as the IP address.

The `uvlictool` also 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. Applies to UniVerse 11.2.2 or earlier.
- On UNIX systems, `uvlictool clean_lic -a` cleans up the current licenses based on shared memory segments associated with dead processes. With this option, `uvlictool` scans the license table and releases any license associated with a pid that is not a valid UniVerse process. Applies to UniVerse 11.2.2 or earlier and 11.3.1 or later.
- On Windows platforms, `uvlictool clean_lic -a` cleans up the current licenses based on dead entries in the process table. With this option, `uvlictool` scans the license table and releases any license associated with a pid that is not a valid UniVerse process. Applies to UniVerse 11.2.2 or earlier and 11.3.1 or later.
- Recomputes license counts at the UniVerse, package, and seat levels. Applies to UniVerse 11.2.2 or earlier.

Syntax

uvlictool [check_acctlic] [list_acctlic] [reload_acctlic] [report_lic [logical_account_name]] [clean_lic [-a]]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
check_acctlic	Checks the syntax of the `$UVHOME/acct_licn.def` file and reports any errors found.
list_acctlic	Lists the current loaded account-based license and its usage.
reload_acctlic	Reloads the `$UVHOME/acct_licn.def` file so that new definitions are applied.
report_lic	Lists the current licensing state. Beginning at UniVerse 11.2.3, the report_lic parameter has been augmented with logical_account_name option. If the logical_account_name is specified, only the users in the specified logical group will be listed.

Parameter	Description
clean_lic	Cleans up the current licensing state. With this option, `uvlictool` scans the license table and releases any license associated with a pid that is not a valid UniVerse process. **Note:** This option was disabled at version 11.2.2, but is re-enabled starting at version 11.3.1. The uvcleanupd daemon performs the same function as the clean_lic option of `uvlictool` when it runs on its regularly scheduled interval.
-a	Recomputes license counts at the UniVerse, package, and seat level. **Note:** This option was disabled at version 11.2.2, but is re-enabled starting at version 11.3.1. The uvcleanupd daemon performs the same function as the clean_lic option of `uvlictool` when it runs on its regularly scheduled interval.

uvlstt

The system-level `uvlstt` command displays details about local control tables (LCTs) in shared memory.

See *Administering UniVerse* for more information about shared memory and LCTs.

Note: This command is available starting at UniVerse 11.3.1.

Syntax

uvlstt [-l *n* |-L*pid*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
-l *n*	Displays additional information about a designated local control table identified by *n*, a local control table.
-L *pid*	Displays additional information about a local control table identified by a *pid*, (a system-level process identification number of a group leader).

Examples

The following example shows general statistical information about all LCTs on a system:

```
# uvlstt
---------------------- LCTs Statistics ----------------------
Total LCTs (Process Groups allowed): 40
LCTs Used (Active Process Groups): 5 (12% of 40)  Total Ps: 10
   Total Global Pages Used: 12 (1536K bytes)
   Total Self-created.....: 0 (0K bytes)
   Total memory used......: 1536K bytes
------------------- End of LCTs Statistics ------------------
:
```

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
prompt	The new command-line-prompt character. The default is > .
select	The new select-list-active character. The default is the same as the command-line-prompt character.
continue	The new line-continue character. The default is + .

UVPROMPT uses only the first character of *prompt*, *select*, and *continue* as the prompt character.

uvregen

The uvregen program verifies that the values in the uvconfig file are reasonable, creates a new .uvconfig file in the UV account directory, and resets the master key if you are using encryption.

Note: Some uvconfig values might be reasonable but invalid for the current kernel configuration. uvregen cannot detect such inconsistencies.

When you make changes to the uvconfig file, you must run the uvregen program and restart UniVerse.

Syntax

uvregen [*options*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
-A *authcode*	Specifies the authorization code to authorize UniVerse, where *authcode* is the authorization code.
-c	Displays the configuration data.
-C	Displays the configuration code.
-d *x*	Specifies the number of connections per device, where *x* should be replaced with actual data.
-e *date*	Specifies an expiration date.
-f	Specifies no warning message or prompt for input (requires -m).
-h	Displays the system help.
-k	Specifies the key reset.

Parameter	Description
-m *enckey*	Specifies UniVerse master encryption key, where *enckey* can be specified as one of the following options: ▪ SYSTEM: Use the system default master key ▪ SYSGEN: Use machine-specific random master key ▪ CURRENT: Set or change the password for the current master key ▪ PROMPT: Let `uvregen` prompt for a master key (secure-input) ▪ <*text*>: Use user-specified text as a master key. Text starting with @ specifies a file containing the master key This option allows only the master key or the password to be changed at one time to avoid the possibility of unintentionally changing either the master key or password due to entering one of the values incorrectly.
-M *key*	Specifies the current UniVerse master key (requires -m).
-o *file*	Stores the SYSGEN-generated key in *file*.
-O *password*	The current master key password (requires -m).
-P *password*	The new master key password to be set (requires -m).
-p *pkg*	Specifies the packages. Use the following format: PKG1:#,PKG2:#,...
-s *n*	Specifies a serial number.
-t	Displays the tunable values.
-T *oldpath,newpath* [-E *file*]	Transfer the master key from an old UVHOME (*oldpath*) to a new UVHOME (*newpath*). Specifying -E *file* outputs the error message to *file* (requires -T).
-u *n*	Specifies the number of UniVerse users.
-z	Displays the license information.

uvrestore

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*] [-noindexfix] [-p] [-P *n*] [-rehash] [-s *file*] [-*startb block*] [{-t *device*} …] [-U] [-v | -V] [-*verify*] [- | *imagepath*][-convtype1]

Options

Specify each option separately, and precede each option with a hyphen.

Parameter	Description
-F	Restores only the specified file. *pathname* must match the file name saved in the image (use the -i option to determine what files are in the image). If you specify *newpathname*, the file originally saved in *pathname* is restored as *newpathname*. To specify more than one file, use multiple -F options.

Parameter	Description	
-R	Restores only the specified records. *record* must match the record ID saved in the image. There can be multiple occurrences of this option. If you specify *newrecord*, the record is written to the file as *newrecord*. You can use the -R option only when you specify one -F option. Any other use terminates the restore process.	
-X	Excludes *pathname* from the restoration. To specify more than one file to exclude, use multiple -X 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. -ib shows the blocks.	
-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.	
-noindexfix	Disables automatic correction of secondary index *pathname*.	
-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 `uvrestore` and return to the shell prompt.
-P	Pauses after *n* 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 *-n* option.	
-s	Specifies a file to which screen output is captured. You must also specify the -v or -V option to use -s.	
-startb	Starts restoring from the block specified by *block*.	
-t	Reads data from *device*. *device* can be either a path or an entry in the &DEVICE& file. Use multiple -t options to specify up to 10 devices. `uvrestore` 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 -U 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 -U, 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.	
-verify	Verifies image integrity.	
-	Takes the name of the restore image from standard input.	
imagepath	Specifies the path of the backup image to restore. *imagepath* can be the name of a device or the name of a file where the image is stored. If you do not specify *imagepath*, `uvrestore` prompts for it.	
-convtype1	When using the `uvbackup` command, UniVerse archives type 1 files as system files. When restoring the file with the -convtype1 option, `uvrestore` remembers all the type 1 files it has restored, and at the end of the process, tries to convert all the type 1 directories to the local format.	

Secondary indexes are automatically restored with the files they index. 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.

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] [-noindexfix] [-p] [-P *n*] [-rehash] [-s *file*] [-startb *block*] [{-t *device*} …] [-U] [-v | -V] [-*verify*] [*imagepath*][-convtype1]

Options

Specify each option separately, and precede each option with a hyphen.

Option	Description
-F	Restores only the specified file. *pathname* must match the file name saved in the image (use the -i option to determine what files are in the image). If you specify *newpathname*, the file originally saved in *pathname* is restored as *newpathname*. To specify more than one file, use multiple -F options.
-R	Restores only the specified records. *record* must match the record ID saved in the image. There can be multiple occurrences of this option. If you specify *newrecord*, the record is written to the file as *newrecord*. You can use the -R option only when you specify one -F option. Any other use terminates the restore process.

Option	Description	
-X	Excludes *pathname* from the restoration. To specify more than one file to exclude, use multiple -X 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. -ib shows the blocks.	
-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.	
-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 `uvrestore` and return to the shell prompt.
-P	Pauses after *n* 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 -n option.	
-s	Specifies a file to which screen output is captured. You must also specify the -v or -V option to use -s.	
-startb	Starts restoring from the block specified by *block*.	
-t	Reads data from *device*. *device* can be either a path or an entry in the &DEVICE& file. Use multiple -t options to specify up to 10 devices. `uvrestore` 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 -U 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 -U, 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.	
-verify	Verifies image integrity.	
imagepath	Specifies the path of the backup image to restore. *imagepath* is the name of a file where the image is stored. If you do not specify *imagepath*, `uvrestore` prompts for it.	

Option	Description
-convtype1	When using the `uvbackup` command, UniVerse archives type 1 files as system files. When restoring the file with the -convtype1 option, `uvrestore` remembers all the type 1 files it has restored, and at the end of the process, tries to convert all the type 1 directories to the local format. When restoring a type 1 file from a Windows platform to a UNIX platform, subdirectories may be created with the same ownership as the parent directory. When restoring a UNIX type 1 file to a Windows platform, these subdirectories are removed.

Secondary indexes are automatically restored with the files they index. 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.

uvsh

Use the operating-system-level `uvsh` command to start a UniVerse session. For UniVerse to run, the product must be installed and licensed.

Syntax

uvsh

uvsms

Use the operating-system-level `uvsms` command to display SMM segment contents.

Syntax

uvsms [*options*]

Options

The following table describes the options available for the `uvsms` command:

Parameter	Description
-h	Displays the SMM segment header.
-G *shm_ID*	Displays the specific shared memory segment you specify with *shm_ID*.
-g *shm_nono*	Displays the specific shared memory segment you specify with *shm_nono*.
-L *pid*	Displays the specific LCT entry you specify with *pid*.
-l *lct_no*	Displays the specific LCT entry you specify with *lct_no*.
-S *shm_ID*	Displays the LCT entry of the session where the shared memory segment is created.

uvtic

`uvtic` compiles the `terminfo` database entries from source into compiled format. UNIX `tic` compiles only the UNIX `terminfo` database, whereas UniVerse `uvtic` compiles both UNIX and UniVerse.

Syntax

uvtic [-a | -v | *file.name*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
-a	**UNIX and Linux only.** Updates the standard UNIX capabilities with those defined in the `terminfo` entries shipped with UniVerse. `uvtic` without the -a option compiles only the UniVerse capabilities into the UniVerse `terminfo` directory located in the UV account directory. If you specify the -a option, `uvtic` compiles the UNIX capabilities into the `/usr/lib/terminfo` and compiles the UniVerse capabilities into the UniVerse `terminfo` directory. The source files are compiled into standard binary file format.
-v	The -v option to `uvtic` allows error messages to be displayed to standard error. If you do not specify -v, unsupported capabilities are ignored. You should be logged on as a UniVerse administrator to execute this command.
file.name	Includes the contents of the terminal type that you want to compile.

uvtidc

`uvtic` decompiles the `terminfo` database entries to produce a source file as a basis for your new `terminfo` definition.

Syntax

uvtidc [-i | *terminal.name*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
-i	Generates the source for UniVerse capabilities.
terminal.name	The name of the terminal.

UV.LOGIN

Use UV.LOGIN to initialize all UniVerse account environments.

Syntax

UV.LOGIN

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 interupt
 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.

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.

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.
filename	The name of an existing file.
records	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.

Example

To edit the program RECEIVABLES in the file BP, enter the following:

```
>UV.VI BP RECEIVABLES
```

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
filename	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 *filename*.O.
catalog	The name of the program in the system catalog space. Catalog names can include only ASCII characters, no multibyte characters.
program	The record in the file *filename*.O 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.

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.

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.

Syntax

VERIFY.DF *dist.filename*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
dist.filename	The name of a distributed file.

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.

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.

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.

Parameter	Description
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.
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.
table	The name of an SQL table or view defined in the VOC file of the current schema.
schema	The name of an SQL schema.
pathname	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.

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.

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 *pathname* FIX VERIFY.SQL SCHEMA 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 TABLE *pathname* FIX	Creates the SQL catalog data using information in the SICA of the table.
VERIFY.SQL VIEW *pathname* FIX	Creates the SQL catalog data using information in the SICA of the view.

Command	Description
VERIFY.SQL SCHEMA *schema* 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 *table* 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.

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 1or type 19 file.

Note: This command is not supported on Windows platforms.

Syntax

VI [*pathname*]

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
pathname	The UNIX path of the file you want to edit. You can use either a relative or an absolute path.

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
filename	The name of the file containing the source code of the BASIC program. The default *filename* is BP.
program	The name of the program to list.

Parameter	Description
R	Specifies to use internal reference numbers for variables and constants rather than the source code names and values.

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 fornext        I to 1 001c

       2: PRINT I
 0002 0010 : 103 printcrlf          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.

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.

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 a key to an encryption wallet.

Syntax

WALLET.ADD.KEY *<wallet_ID> <wallet_password> <key_ID> <key_password>*

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
wallet_ID	The ID of the wallet to which you want to add a key.
wallet_password	The password for the wallet to which you want to add a key.
key_ID	The key_ID you want to add to the wallet.
key_password	The password for the key you want to add to the wallet. **Note:** Keys that are entered in a wallet must have a password.

WALLET.REMOVE.KEY

Use the WALLET.REMOVE.KEY to remove an encryption key from a 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 remove an encryption key.
wallet_password	The password for the encryption key you want to remove from the encryption wallet.
key_ID	The key_ID you want to remove from the encryption wallet.
key_password	The password from the key you want to remove from the encryption wallet.

WARNINGS

The `WARNINGS` command determines if UniVerse displays warning messages when running a UniVerse BASIC program.

Syntax

`WARNINGS` `[ON | OFF]`

Parameters

The following table describes each parameter of the syntax.

Parameter	Description
ON	Displays warning messages that occur when running a UniVerse BASIC program. This is the default.
OFF	Suppresses all warning (nonfatal) error messages.

If you run the `WARNINGS` command with no parameters, UniVerse displays the current status of the `WARNINGS` flag, as shown in the following example:

```
>WARNINGS
Run time warning turned ON
```

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*.

Chapter 2: UniVerse keywords

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

Symbol	Synonyms	Description
%	PCT PERCENT PERCENTAGE	Calculate percentages
&	AND	The logical operator AND
(		Open parenthesis
)		Close parenthesis

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.
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 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.

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.

Keyword	Synonym	Description
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.
NO.PAGE	NO.PAGE	Suppresses automatic paging.
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.

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 SET.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:

```
ASSOC.WITH {fieldname | 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 reports are created in &HOLD&. If you do not specify *name*, the record ID is P#0000_*nnnn*, where *nnnn* is increased for each new print job. If you specify *name*, the record ID is name_*nnnn*.

In mode 3, BANNER UNIQUE appends a sequential number to the name under which successive records are created in &HOLD&. If you do not specify *name*, the record ID is P#0000_*nnnn*, where *nnnn* is increased by each subsequent `SETPTR` command. If you specify *name*, the record ID is *name_nnnn*.

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.

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.

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 first exploded field does not belong to any association, RetrieVe assumes a default association for the duration of the query. This does not apply to additional exploded fields which must be explicitly associated. 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 first exploded field does not belong to any association, RetrieVe assumes a default association for the duration of the query. This does not apply to additional exploded fields which must be explicitly associated.

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.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 or LIST.INTER 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 multivalue 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.id*V*value*S*field*V0

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 " [text] ['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 *n* 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 *n* 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 *n* spaces.
G	Inserts gaps in the footing format.
I[*n*]	Inserts the record ID left-justified in a field of *n* spaces. Same as R.

Option	Description
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 *n* 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 *n* 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

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"	| Hello there|
"'G'Hello there'G'"	| Hello there |
"Hello'G'there"	|Hello there|
"'G'Hello'G'there'G'"	| Hello there |

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.

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 to specify the size of each group in a dynamic (type 30) file.

Its syntax is as follows:

```
GROUP.SIZE {n}
```

where *n* is the separation, which is a multiple of 2K of the value entered. The default is 1 for a separation of 2K.

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 *n* 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 *n* 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 *n* spaces.
G	Inserts gaps in the heading format.
I[*n*]	Inserts the record ID left-justified in a field of *n* 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.

Option	Description
P[*n*]	Inserts the page number left-justified in a field of *n* 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 *n* 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

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"	| Hello there|
"'G'Hello there'G'"	| Hello there |
"Hello'G'there"	|Hello there|
"'G'Hello'G'there'G'"	| Hello there |

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 (UNIX).

Use with the SPOOL 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:

```
{WITH | WHEN | IF} 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:

```
{WITH | WHEN | IF} 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, 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).
*n*X	*n* number of any characters.
0A	Any number of alphabetic characters (including none).
*n*A	*n* number of alphabetic characters.
0N	Any number of numeric characters (including none).
*n*N	*n* number of numeric characters.
'*text*'	Exact text; any literal string (quotation marks required).
"*text*"	Exact text; any literal string (quotation marks required).

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
string …	Matches records with a field beginning with *string.*
… *string* …	Matches records with *string* anywhere in the field.

Syntax	Description
... string	Matches records with a field ending with *string*.
string1 ... string2	Matches records with a field beginning with *string1*, ending with *string2*, and containing any number of characters between them.

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.

The following example displays the output of this option:

```
>PORT.STATUS PID 11403388 MFILE.HIST
There are currently 1 uniVerse sessions; 1 interactive, 0
phantom
Pid....  User name.  Who.  Port name.....  Last command processed.............
11403388  root          11  /dev/pts/2      SH
-c "printenv UDTHOME" [ *AE_AE @ 0x5AC ]

File name.....................................  File No   Chan
VOC                                               3       5
/disk1/uv/SYS.MESSAGE                             1       4
CUSTOMER/DATA.30                                  16      9
VOC                                               13      6
AE_COMS                                           14      7
/disk1/uv/AE_DOC

Action code... File    Caller...
MFcheck         14     DBread
MFcheck         14     DBread
MFcheck         14     DBread
MFclose         21     DBclose
MFclose         20     DBclose
MFclose         19     DBclose
MFclose         17     DBclose
MFclose         16     DBclose
MFclose         15     DBclose
MFclose         23     DBclose
MFclose         22     DBclose
```

```
MFcheck         1       DBread
```

The following table describes the columns of the MFILE.HIST output.

Column	Description
File No	The file descriptor seen at the operating system level.
Chan	The MFILES slot for the file. -1 indicates that the file is logically opened in UniVerse, but closed at the operating system level.
Action code	The name of the internal UniVerse low-level function which operated with the corresponding file descriptor.
File	The file descriptor number actioned by the corresponding internal functions. Note that this file number could no longer appear as an active file if it was closed prior to the `PORT.STATUS PID xxxxx MFILE.HIST` command performed.
Caller	The name of the internal UniVerse high-level function that used the file descriptor for the corresponding low-level operation.

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. RetrieVe treats nonnumerics as zero values.

NO.NULLS tells RetrieVe to ignore empty string values.

The minimum value is the same as the first value that appears when you use a RetrieVe 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, RetrieVe 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, RetrieVe treats each value separately.

If you use the MIN keyword with a breakpoint keyword (BREAK.ON or BREAK.SUP), RetrieVe 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
m	Mode	0 - ASCII, no conversions 1 - EBCDIC conversion 2 - Invert high-order bit 3 - Invert high-order bit and EBCDIC conversion
t	Tracks	Not used
u	Unit number	Any number from 0 through 7 indicating the tape drive number

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).

Pattern	Definition
0X	Any number of any characters (including none).
*n*X	*n* number of any characters.
0A	Any number of alphabetic characters (including none).
*n*A	*n* number of alphabetic characters.
0N	Any number of numeric characters (including none).
*n*N	*n* number of numeric characters.
'*text*'	Exact text; any literal string (quotation marks required).
"*text*"	Exact text; any literal string (quotation marks required).

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
string …	Matches records with a field beginning with *string.*
… string …	Matches records with *string* anywhere in the field.
… string	Matches records with a field ending with *string*.
string1 … string2	Matches records with a field beginning with *string1*, ending with *string2*, and with any number of characters between them.

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.INDEXto 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 SET.FILE 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 "" xxxx "" HH:MM:SS

"" DD "" MMM "" YYYY

"" filename "" header pad F rr

PICK.FORMAT	
I	Item mark (segment mark)
L	ASCII character L
xxxx	Tape record size (hexadecimal)

PICK.FORMAT	
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)

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, I represents an item mark, F represents a field mark, and V represents a value mark):

I L header V HH:MM:SS

"" DD "" MMM "" YYYY

F rr F xxxxx F I pad

REALITY.FORMAT	
I	Item mark (segment mark)
L	ASCII character L
header	(Optional) Label header text (<= 44 characters)
V	Value mark (follows L if no header)
HH:MM:SS	Time (external 24-hour)
DD MMM YYYY	Date (external form)
F	Field mark (attribute mark)
rr	Reel number (decimal)
xxxxx	Tape record size (decimal)
pad	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.DUE1 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 RetrieVe sentence to make it more English-like. This keyword is not required and is ignored by RetrieVe.

THE

Use in any RetrieVe sentence to make it more English-like. This keyword is not required and is ignored by RetrieVe.

THEN

Use in any RetrieVe sentence to make it more English-like. This keyword is not required and is ignored by RetrieVe.

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.DATE2/1/92 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 DIVERT.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 DIVERT.OUT to turn off terminal (tty) display of diverted output. User input is also turned off.

TTY.ON

Use with DIVERT.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.DUE1 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
field	If you use a field name, ReVise verifies that the value entered in *verifield* is also a record ID in *verifile*.
record.ID	If you use the record ID field name, ReVise verifies that the entered value is not a record ID in *verifile*. If the ReVise sentence also includes a VERIFIELD expression, ReVise verifies that the record ID entered is a record ID in *verifile* and displays the value of the VERIFIELD display field.

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 mvfield operator {values | field2 }
[[{AND | OR} mvfield operator] {values | field2} …]
```

mvfield 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 *mvfield*. If you specify more than one value, the OR operator is assumed; that is, the condition:

mvfield operator value1 value2

means:

*mvfield operator value1*OR *mvfield 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 *mvfield*.

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 mvfield operator value   WHEN mvfield operator
value
```

means:

```
LIST filename WHEN mvfield operator value   AND WHEN mvfield 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.DUE0 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.DUE50 AND (PYMT.DATE < 2/1/95 OR
PYMT.DATE2/28/95)
```

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
op value	WITH @ID *op value*
WITH *field*	WITH *field* NE ""
WITH *field1 field2*	WITH *field1* = *field2*
WITH NOT *field*	WITH *field* = ""
WITH *field op value* OR *op value*	WITH *field op value* OR *field op value*

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 ZIP44000 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.

Chapter 3: User records

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.

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.

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.

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.

You can set and display the Quit key with the PTERM (UNIX) or the PTERM (Windows platforms) command.

Note: UniVerse loads the values for QUIT.KEY 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.

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.

Chapter 4: Local and system files

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

&COMO&

&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&

&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           5R     S
                                                size
TYPE           D    4                          Device          2L     S
                                                type
LOCK           D    5                          Lock files      25L    M
REWDEV         D    6                          Rewind          25L    S
                                                device
NREWDEV        D    7                          NoRewind        25L    S
                                                device
BACKUP         D    8                          cpio-backup     50L    S
RESTORE        D    9                          cpio-restore    50L    S
SKIP           D   10                          Skip            50L    S
                                                file
REWIND         D   11                          Rewind          50L    S
                                                tape
OFFLINE        D   12                          Tape            50L    S
                                                offline
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           6R     S
                                                Blksize
ROTFLG         D   17                          Rotate          1L     S
                                                Flag
OTHER          D   18                          Field 18        70T    M
NLSMAP         D   19                          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&

&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 *name*	Deletes *name* from &ED&.
.L *name*	Lists *name* from &ED&.
.R *name*	Loads *name* from &ED& into the current Editor command stack.
.S *name s e*	Saves lines *s* through *e* from the command stack as *name* in &ED&.
.S *n name*	Saves lines 1 through *n* from the command stack as *name* in &ED&.
.X *name*	Executes the saved command record *name* from &ED&.

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&

&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&

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/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&

&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&

&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&

&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&

&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&

&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&

&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

APP.PROGS is a type 1 system file used to store the source of programs that U2 neither supports nor warrants. The object code for these programs is stored in the corresponding type 1 file, APP.PROGS.O.

BLTRS

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]1 V 2 V 3 V 4 V 5, 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

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

DICT.PICK is a system file dictionary that functions as a master file dictionary for RetrieVe processing of all the Pick-style dictionaries.

NEWACC

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

REVISE.PROCESSES is a hashed system file that contains the templates used by ReVise in building dictionaries, menus, and selectors for menus.

STAT.FILE

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

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, on page 482.

SYS.MESSAGE

SYS.MESSAGE is a hashed system file. The records in the SYS.MESSAGE file contain the UniVerse system messages.

UNIVERSE.MENU.FILE

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 2Creating file "MENU.FILE" as Type 3, Modulo 1, Separation 2.
 >ED VOC MENU.FILE3 lines long.
----: 30003:
 Bottom at line 3.
 ----: DBottom 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

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, on page 480.

UNIVERSE.VOCLIB

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

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.

Appendix A: UniVerse commands quick reference

This appendix briefly describes the UniVerse commands described in detail in UniVerse commands, on page 18.

The commands are grouped according to use as follows. Command descriptions can also be found online in the VOC file.

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.

Command	Description
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.
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.
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)	Begins a daily, weekly, or full save of an entire system, or selected directories or files.
uvbackup (Windows platforms)	Restores an entire system, or selected directories, files, or records in UniVerse hashed files saved by *uvbackup*.
WHO	Displays the port number and account name.

Parent topic: UniVerse commands quick reference

Arithmetic commands

Command	Description
PRIME	Finds the nearest prime numbers.
RADIX	Converts the number base.

Parent topic: UniVerse commands quick reference

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.

Command	Description
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.
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.

Parent topic: UniVerse commands quick reference

Managing BASIC programs

Command	Description
BASIC	Compiles a BASIC program.
CATALOG	Catalogs a compiled BASIC program.
CLEARCOMMON	Resets theBASIC 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.

Parent topic: UniVerse commands quick reference

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.

Command	Description
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.

Parent topic: UniVerse commands quick reference

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.

Parent topic: UniVerse commands quick reference

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 *vi* editor using the file naming syntax. This command is not supported on Windows platforms.
VI	Invokes the UNIX *vi* editor. This command is not supported on Windows platforms.

Parent topic: UniVerse commands quick reference

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.

Parent topic: UniVerse commands quick reference

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 contained in UniVerse shared memory.
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 (*csh*). 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.
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.

Command	Description
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 `Press any key to continue...` 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 (*sh*). 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.
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 uvrestore (Windows platforms)	Restores an entire system, or selected directories, files, or records in UniVerse hashed files saved by *uvbackup*.
WHO	Displays the port number and account name.

Parent topic: UniVerse commands quick reference

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.
LOG.RESTORE	Restores transaction logging log files from tape to disk.

Command	Description
LOG.SAVE	Saves transaction logging files from disk to tape.
MKFILE.LIST	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.LOG.ATTR	Sets or changes the transaction logging operating mode.
SHUTDOWN.RECOVERY	Disables the transaction logging system.
SUSPEND.RECOVERY	Suspends the transaction logging system.

Parent topic: UniVerse commands quick reference

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.
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.
VERIFY.DF	Verifies the paths and part numbers of part files belonging to a distributed file.
VERIFY.SQL	Verifies and fixes SQL catalog inconsistencies.

Parent topic: UniVerse commands quick reference

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.

Parent topic: UniVerse commands quick reference

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.

Parent topic: UniVerse commands quick reference

Paragraph commands

Command	Description
.S	A sentence stack command that saves sentences in a VOC file entry.
*	(Asterisk) Indicates that the text that follows is part of a comment.
<< … >>	Displays a prompt and requests an input value when a sentence is executed.
CLEARDATA	Clears a data stack.
CLEARPROMPTS	Sets the value of in-line prompts to empty strings.
DATA	Specifies a response to input requests.
DISPLAY	Prints a line on the terminal.
GO	Transfers execution to a subsequent sentence in a paragraph.
IF	Conditionally executes statements in a paragraph.
LOOP	Defines the beginning of a range of sentences to be repeatedly executed.
REPEAT	Ends a LOOP…REPEAT loop in a paragraph.

Parent topic: UniVerse commands quick reference

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.

Command	Description
usa	Administers the UniVerse spooler from a UNIX shell. This command is not supported on Windows platforms.
usd	Starts the UniVerse spooler from a UNIX shell. This command is not supported on Windows platforms.
usm	Changes print job characteristics from a UNIX shell. This command is not supported on Windows platforms.
usp	Sends print jobs to the UniVerse spooler from a UNIX shell. This command is not supported on Windows platforms.

Parent topic: UniVerse commands quick reference

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.

Parent topic: UniVerse commands quick reference

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.

Command	Description
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.

Parent topic: UniVerse commands quick reference

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.

Parent topic: UniVerse commands quick reference

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.

Command	Description
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.

Parent topic: UniVerse commands quick reference

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) PTERM (Windows platforms)	Sets and displays terminal characteristics.
SET.TERM.TYPE	Sets the terminal type.
TERM	Sets printer and terminal characteristics.
TERM	Displays the port number and account name.

Parent topic: UniVerse commands quick reference

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.

Command	Description
LISTPA	Lists paragraphs stored in the VOC file.
LISTPH	Lists phrases stored in the VOC file.
LISTQ	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.
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.

Parent topic: UniVerse commands quick reference

Appendix B: The XDEMO account

Starting at UniVerse 11.3.1, you can use the XDEMO account to test and use UniVerse commands. The XDEMO account is a group of test files that are installed automatically with UniVerse on Windows. For UNIX platforms, the XDEMO account is optional.

Note: XDEMO does not include any indexes; only test programs and files are included.

For more information about the installation of XDEMO, see the *Installation Guide*.

XDEMO files

The following table describes the files in the XDEMO account in more detail.

File	Description
FUR_REV	Contains 512 records, it includes an example revenue file.
LOCATIONS	Contains 3 records. Addresses, phone and fax numbers are not real numbers.
MEMBERS	Contains 2500 records. It includes imaginary credit card information and other data. The data is generated randomly. For the PASSWORD and CARDNUM fields, there is an additional ENC.PASSWORD and ENC.CARDNUM field that contain encrypted versions of those fields. In the dictionary of the MEMBERS file, a record called ENC.PARAMS reports which parameters were used for these fields. In the XDBP directory, there is a DECRYPT.EXAMPLE program that shows how the data can be decrypted again for passwords and credit card numbers. Records in this file contain no real life data.
PLOCATION	Contains 57399 records. This includes all delivery pizza locations in the USA. It includes longitude and latitude positions for use with mapping software.
PP & PBP	Contains U2 Python code examples, PP is just Python code and PBP is BASIC code just for Python examples.
PRODUCT_REVIEW	Contains 2 records only.
PRODUCTS	Contains 390 records. There are no duplicate editions of titles, and only movies that have a synopsis are included.
REGION	Contains about 3 records, all related to the PLOCATION file.
RENTAL_DETAILS	Contains 2240 records. The dictionary items have similar syntax.
STATES	Includes a REGION and DIVISION field.
STATES_MAPS	Contains 54 records. Used in conjunction with PLOCATION.
XDBP	Basic programs used in examples. **Note:** The programs in the XDBP/XDBP.O files are cataloged using the DIRECT option on UniData and LOCAL on UniVerse.
WORLD_MAP	Contains 239 records. Related to PLOCATION file.
ZIPCODES	Contains 100 records. Related to PLOCATION file.