



## **LaserVault Reports Indexing Guide**

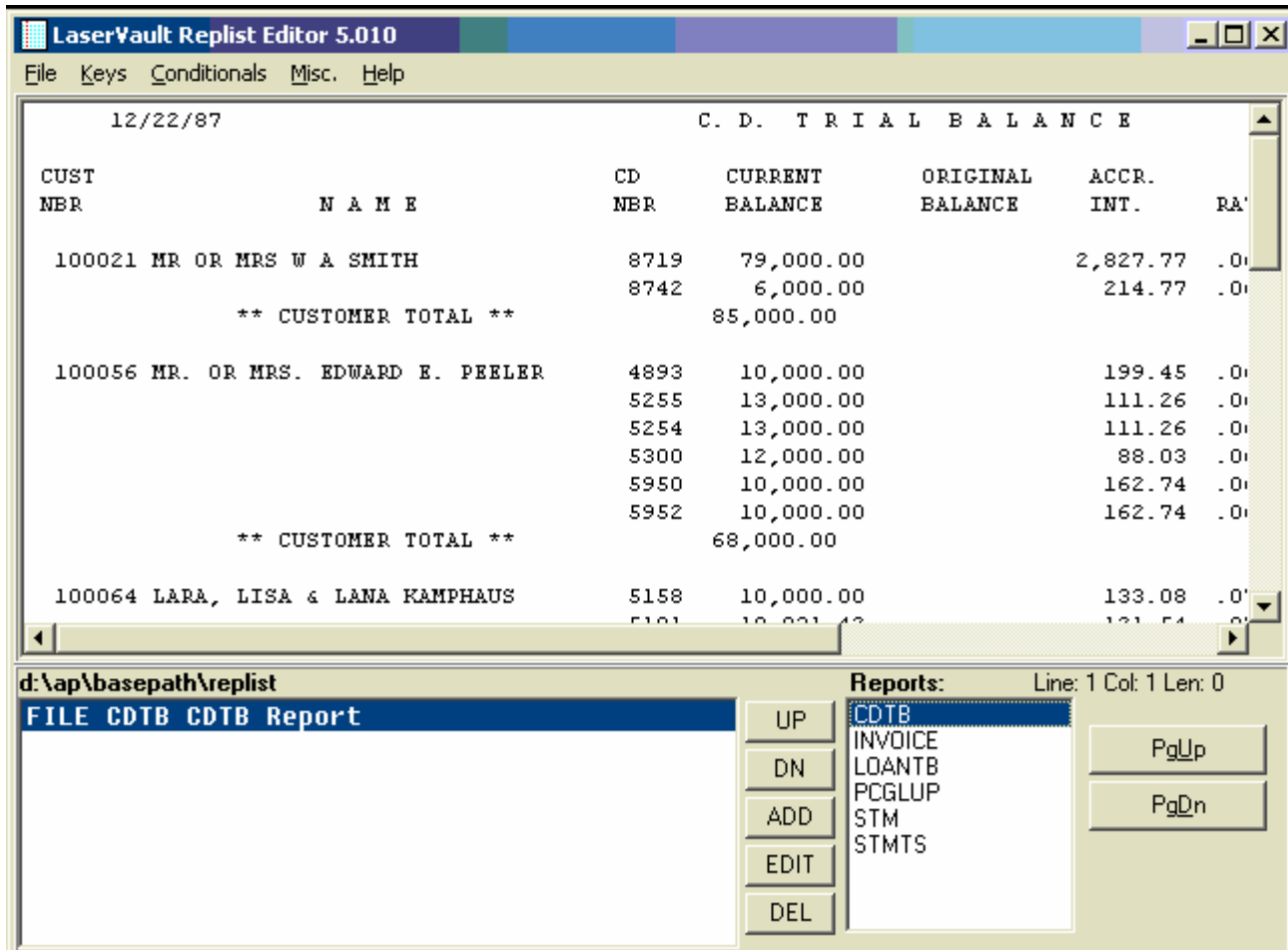
<http://www.laservault.com>

# LaserVault Replist Editor

The LaserVault Replist (Report List) Editor is a utility that helps you define index keys on reports. The replist itself is a text file and can be edited with any text editor. The replist editor makes defining keys easier by displaying the report and allowing you to highlight text to define the keys.

Start the Replist Editor and open the replist file in the base path of the archive you want to work in. You can also launch the replist editor from the archives screen in the management console.

Make sure you have the reports you want to define keys on in the base path.



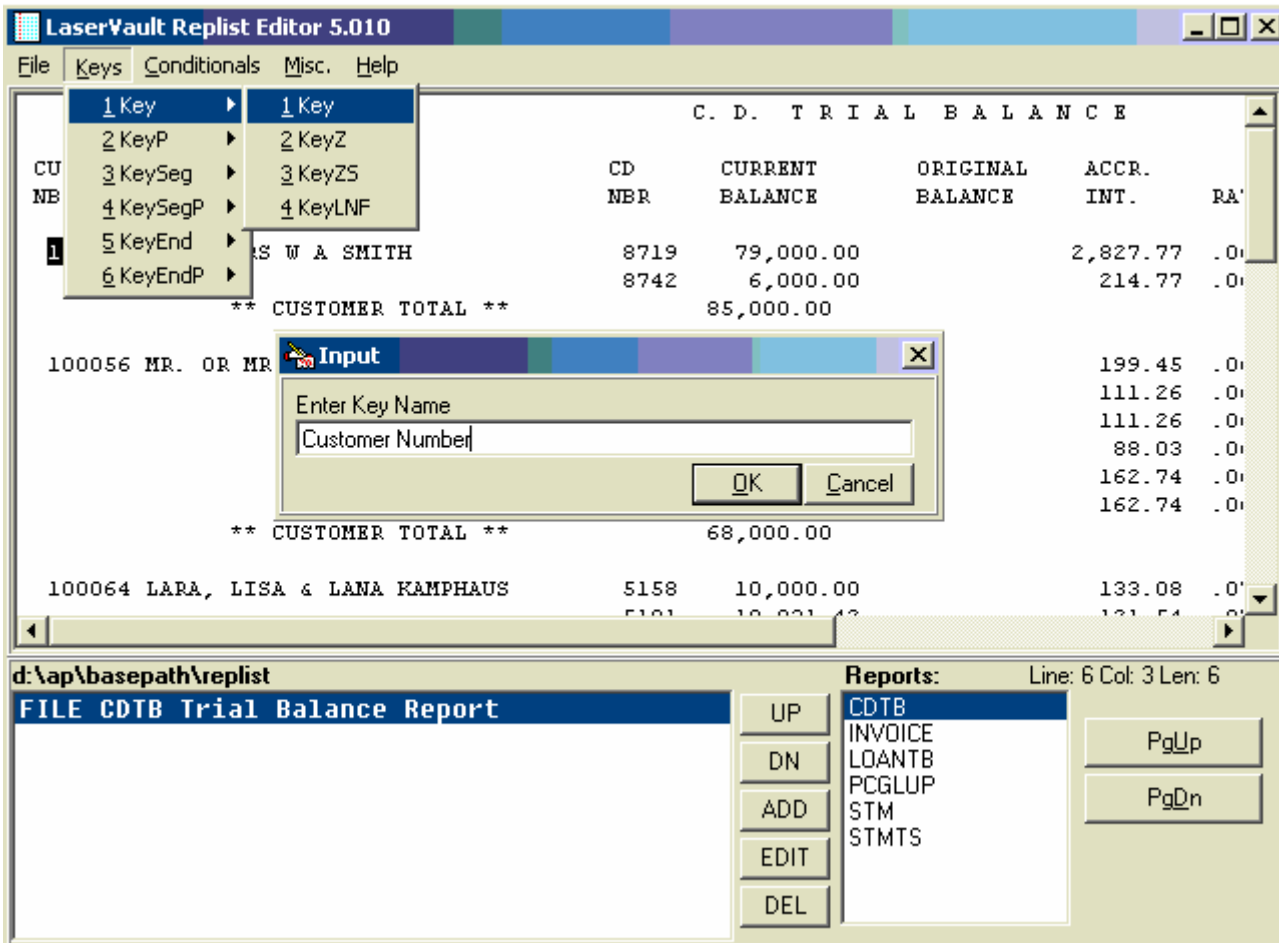
After starting the replist editor and selecting the replist file in the basepath, select the report in the list. You will see the text of the report displayed in the top window. You will also see the default "File" statement in the left window. You can edit this line and change the description by clicking the edit button.

The screenshot shows the LaserVault Replist Editor 5.010 interface. The main window displays a trial balance report for 12/22/87. The report is titled "C. D. T R I A L B A L A N C E" and lists customer accounts with their current and original balances, accrued interest, and payment amounts. An "Input" dialog box is open over the report, titled "Edit Replist Entry", with a text field containing "Trial Balance Report" and "OK" and "Cancel" buttons.

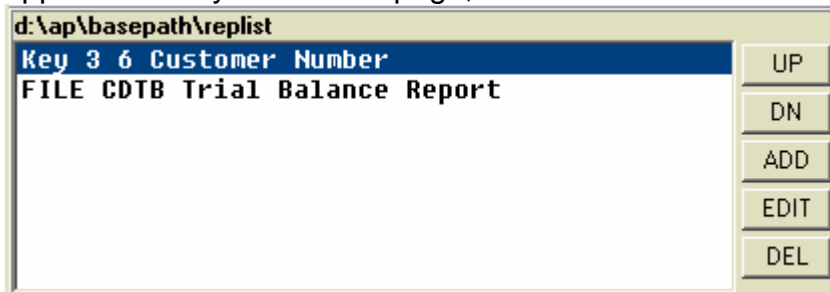
CUST NBR	N A M E	CD NBR	CURRENT BALANCE	ORIGINAL BALANCE	ACCR. INT.	PA
100021	MR OR MRS W A SMITH	8719	79,000.00		2,827.77	.00
		8742	6,000.00		214.77	.00
** CUSTOMER TOTAL **			85,000.00			
100056	MR. OR MRS. EDWARD E. PEELER	4893	10,000.00		199.45	.00
					111.26	.00
					111.26	.00
					88.03	.00
					162.74	.00
					162.74	.00
100064	LARA, LISA & LANA KAMPHAUS	5158	10,000.00		133.08	.00
		5181	10,000.00		131.54	.00

The bottom panel shows the file path "d:\ap\basepath\replist" and a list of reports: "FILE CDTB CDTB Report", "CDTB", "INVOICE", "LOANTB", "PCGLUP", "STM", and "STMTS". Navigation buttons include UP, DN, ADD, EDIT, DEL, PgUp, and PgDn. The status bar indicates "Line: 1 Col: 1 Len: 0".

To define an index key, highlight the text you want to build the key on and click the keys menu. Select Key and then enter the key name.



The replis editor added a line defining a key that starts at column 3, has a length of 6 characters, applies to every line on the page, and has a name of "Customer Number".



To apply criteria for a key, add a "WhenCol" statement.

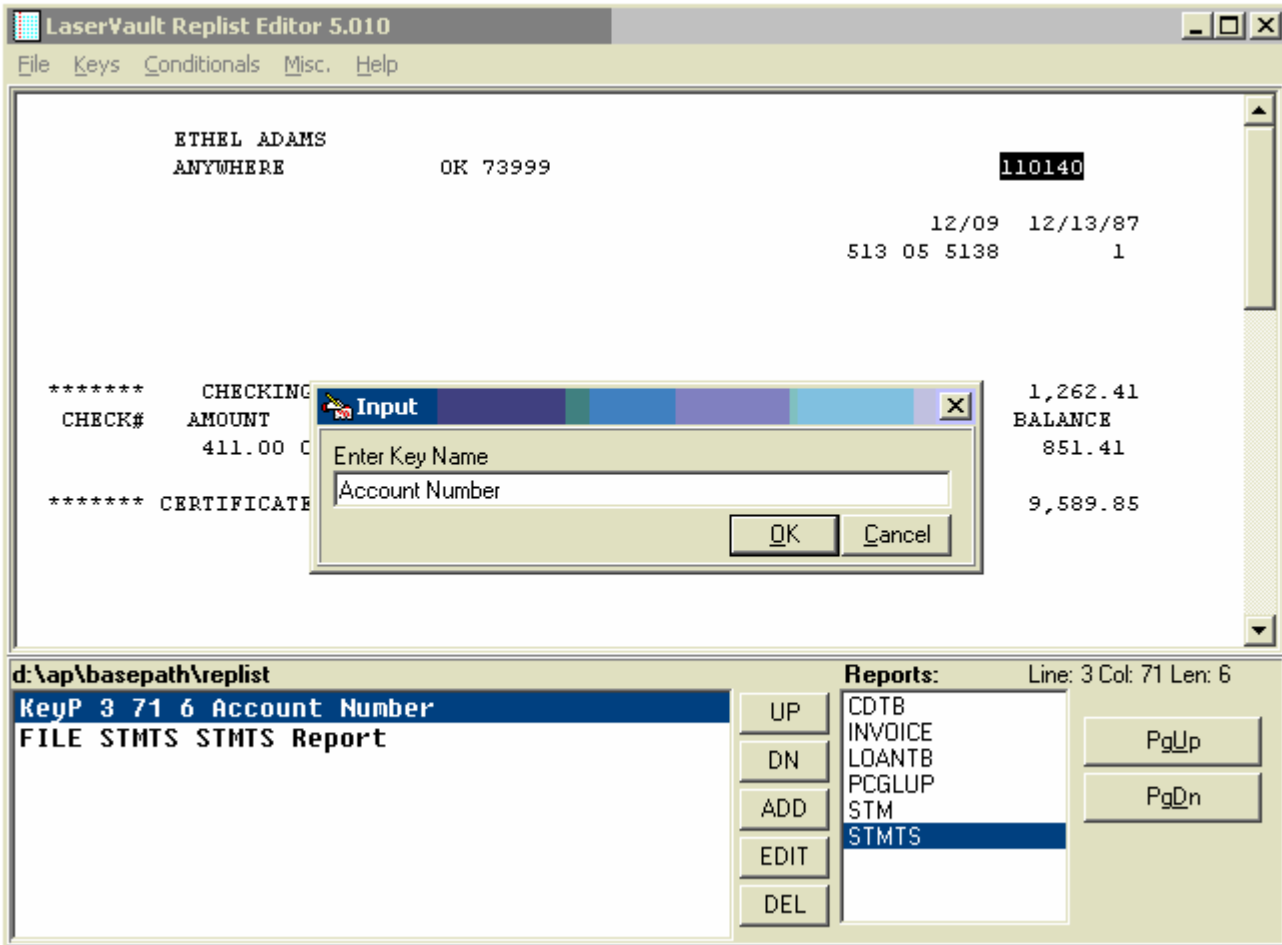
The screenshot shows the LaserVault Replist Editor 5.010 interface. The main window displays a trial balance report for 12/22/87. The report is organized into columns: CUST NBR, NAME, CD NBR, CURRENT BALANCE, ORIGINAL BALANCE, ACCR. INT., and PA. The data is grouped by customer, with totals for each group.

CUST NBR	NAME	CD NBR	CURRENT BALANCE	ORIGINAL BALANCE	ACCR. INT.	PA.
100021	MR OR MRS W A SMITH	8719	79,000.00		2,827.77	.00
		8742	6,000.00		214.77	.00
** CUSTOMER TOTAL **			85,000.00			
100056	MR. OR MRS. EDWARD E. PEELER	4893	10,000.00		199.45	.00
		5255	13,000.00		111.26	.00
		5254	13,000.00		111.26	.00
		5300	12,000.00		88.03	.00
		5950	10,000.00		162.74	.00
		5952	10,000.00		162.74	.00
** CUSTOMER TOTAL **			68,000.00			
100064	LARA, LISA & LANA K&MPHAUS	5158	10,000.00		133.08	.00

The bottom panel of the editor shows the key configuration for the 'CDTB Trial Balance Report'. It includes a 'WhenCol' statement: 'WhenCol 58 "."', which specifies that the key is only built when column 58 contains a decimal character. The key is defined as 'Key 3 6 Customer Number'. The 'Reports' list includes CDTB, INVOICE, LOANTB, PCGLUP, STM, and STMTS. Navigation buttons (UP, DN, ADD, EDIT, DEL) and page navigation buttons (PgUp, PgDn) are also visible.

What the whencol statement does is only builds a key when column "x" contains the specified literal value. In the example above when column 58 contains the decimal character, then build the customer name key. The whencol statement applies to any key defined after it in the list. This can help you avoid getting header or other unwanted data in the index.

To define a key that only occurs once per page, use the KEYP statement.



In this example, we highlighted the value we wanted for the index then selected KEYP. This statement defines a line number, column, number of characters then the key name. In this example we are using the value on line 3, column 71, and length of 6 characters for the account number key.

After defining keys, save the replist and exit the replist editor. Then index the reports in the base path and use LaserVault to view the reports and test your key definitions.

# Replisr Statements

## **KEY**

This defines the location on every line to extract data for a key value to be inserted into the index file. There are 3 parameters - beginning column, key length, and up to 20 characters of key description.

Example: KEY 2 6 Customer#

## **KEYP**

This defines the location on a specific line to extract data for a key value to be inserted into the index file.

There are 4 parameters - beginning line#, beginning column, key length, and key description. Only one key value will be extracted per report page.

Example: KEYP 4 2 6 Customer#

## **WHENCOL**

This is used to build a condition which must be proven true in order for a key value to be extracted for following KEY and KEYP statements. There are 2 parameters for this statement, the beginning column and string for a comparison. The case of the characters must match that of the string being checked for in the report.

Use double quotes (") when the string has imbedded blanks within it. A value of 999 for the beginning column will look for the comparison to occur anywhere on the line. A WHENCOL statement remains in effect until another WHENCOL, a WHENNOTCOL, a CLEARFLAGS, or a FILE statement is encountered.

Tip: Some common things to check for are decimal points in dollar amounts and slashes in dates.

Example: WHENCOL 101 .  
WHENCOL 1 "Customer Number:"

## **WHENNOTCOL**

This is used the same as WHENCOL, except it is a negative.

## **FILE**

This defines the report file that all of the preceding statements apply to. There are 2 parameters for this statement, the DOS file name, and up to 40 characters of description for the report to be stored for LV to use. The DOS file name is only the 8 character base name, the file must have a .RPT for the extension for LV to recognize it.

Example: FILE WKINV Weekly Invoices

### ***NOREPEAT***

This is used to specify that only one key value will be extracted for a key that appears multiple times consecutively on the same page. If the same key value continues consecutively across one or more page breaks, it will appear once for each page it is on.

Example: NOREPEAT  
KEY 2 6 Customer#

### ***NOREPEATEXT***

This keyword functions the same as NOREPEAT shown above with the exception that if the same key value continues consecutively across one or more page breaks, it will not cause an index entry to be created for each page it is on, only for the very first occurrence.

Example: NOREPEATEXT  
KEY 2 6 Customer#

### ***CLEARFLAGS***

This is used to clear the effects of any preceding modifier statements such as NOREPEAT, AFTERLINE, ONLYLINE, WHENCOL, WHENNOTCOL.

Example: CLEARFLAGS

### ***ANDCOL, ORCOL, ANDNOTCOL, ORNOTCOL***

These are used in addition to WHENCOL, and/or WHENNOTCOL. Only one ANDCOL, ORCOL, ANDNOTCOL, or ORNOTCOL may be used with each WHENCOL or WHENNOTCOL.

Caution should be exercised when using ORNOTCOL with WHENCOL, or ORCOL with WHENNOTCOL due to complexities of logic when using a NOT condition in an OR situation (in other words you may not properly calculate all of the possibilities of the given logic).

### ***AFTERLINE***

This is used to specify a line# in which to begin checking for key extractions after. There is only one parameter which is the line#.

Example: AFTERLINE 6  
KEY 2 6 Customer#

### ***ONLYLINE***

This is used to specify that following KEYP statements are only to extract key values found on the beginning line# specified on the KEYP statement. Normally, a KEYP statement will only specify a beginning line# which means that if that position on the beginning line is blank, successive lines will be examined for key extraction until a value is found for the key. There are no parameters for this.

Example: ONLYLINE  
KEYP 4 2 6 Customer#

### **KEYSEG, KEYEND**

These two statements are used in conjunction with each other. These are used to define a key which is made of multiple segments or values. The parameters are the same as the KEY statement. NOTE: During a key lookup, LV will position to the last portion of the key segment.

Restrictions: Each segment must appear on the same line as or a previous line of the next segment of the key. The key description of the first KEYSEG is the description stored for LV to use. The key description of each successive KEYSEG and of the KEYEND must be a plus sign (+) or at least begin with one. Each KEYSEG and KEYEND count toward the limitation of 9 keys per report.

Example:

```
KEYSEG 4 6 Cust#/Inv#/WO#
KEYSEG 12 7 +Invoice#
KEYEND 54 7 +Work Order#
```

Example:

```
KEYSEG 4 6 Cust#/Inv#/WO#
KEYSEG 12 7 +
KEYEND 54 7 +
```

### **KEYSEGP, KEYENDP**

These two statements are used the same as KEYSEG and KEYEND, and may be use in conjunction with KEYSEG and KEYEND. The parameters are the same as the KEYP statement. NOTE: During a key lookup, LV will position to the last portion of the key segment.

Example:

```
KEYSEGP 2 4 6 Cust#/Inv#/WO#
KEYSEGP 3 12 7 +Invoice#
KEYENDP 3 54 7 +Work Order#
```

Example:

```
KEYSEGP 2 4 6 Cust#/Inv#/WO#
KEYSEGP 3 12 7 +
KEYENDP 3 54 7 +
```

### **KEYSEP**

This keyword is used in conjunction with the variations of the KEYSEG and KEYEND keywords. The character specified for the value of this keyword is used as a seperator between key segment values. The key values are trimmed to remove any leading and trailing blanks with the seperator character replacing them. In the example given below, if Cust# has a value of 6420, Inv# has a value of 345698 and Work Order# has a value of 4832, the user would key in 6420/345698/4832 to find the position in the report corresponding with that key. Without the KEYSEP keyword in the example below, given the same segment values shown above the user would have to key in 6420 345698 4832 in order to find the position in the report corresponding with that key.

Example:

```
KEYSEP /  
KEYSEGP 2 4 6 Cust#/Inv#/WO#  
KEYSEGP 3 12 7 +Invoice#  
KEYENDP 3 54 7 +Work Order#
```

### **KEYZ**

This is the same as the KEY statement except that during the indexing process, key values that begin with blanks will be zero-filled instead of being left-justified as normally done. The contents of the report are not changed, only the values inserted into the index file are modified.

For example, if zero-suppression was used during printing of a report, a value of 10 in a 6 digit output would appear as " 10". This would normally be stored in the index file as "10 ". Using the KEYZ statement, however, would store the value as "000010". Users should be made aware of this fact, because to retrieve the proper key value they would have to insert the zeros on the key prompt.

Example: KEYZ 2 6 Customer#

### **KEYZS**

This is the same as the KEY statement except that during the indexing process, key values that begin with zeros will be zero-suppressed before being left-justified. The contents of the report are not changed, only the values inserted into the index file are modified.

For example, if the value of 10 was printed on a report in a 6 digit output without zero-suppression, it would appear as "000010". This would normally be stored in the index file as "000010". Using the KEYZS statement, however, would store the value as "10". Users should be made aware of this fact, because to retrieve the proper key value they would have to omit the leading zeros on the key prompt. This would also affect the get-next-key value operation in LaserVault in a manner that the value 10 instead of the value 2 would follow 1 since the leading zeros are not present in the key value any longer.

Example: KEYZS 2 6 Customer#

### **KEYLNF**

This is the same as the KEY statement except that during the indexing process, the key is assumed to be a person's name and the last name is inserted first into the key value. The contents of the report are not changed, only the values inserted into the index file are modified.

Restrictions: The process used to extract the last name from the key value is simply to scan the key value from the end until a blank is encountered preceding the last name. This will result in incorrect key values for those people who have complex last names such as St. John which would result in only John being moved to the first of the key value. A solution to this problem would be to change the entry of the name to St.John without the blank.

Another example of incorrect key values would be the use of this on reports where names of people are mixed with names of companies. The company names would also be reversed.

Example: KEYLNF 2 25 Customer Name

***KEYPN, KEYPX, KEYPZ, KEYPLNF***

These are the variations of KEYP which function the same as the variations KEYN, KEYX, KEYZ, and KEYLNF for the KEY statement.

***KEYSEGN, KEYSEGX, KEYSEGZ, KEYSEGLNF***

***KEYENDN, KEYENDX, KEYENDZ, KEYENDLNF***

These are the variations of KEYSEG, and KEYEND.

***KEYSEGPN, KEYSEGPX, KEYSEGPZ, KEYSEGPLNF***

***KEYENDPN, KEYENDPX, KEYENDPZ, KEYENDPLNF***

These are the variations of KEYSEGP, and KEYENDP.