

# Replist Editor Unleashed

The Replist file consists of commands that “tell” the indexing program how to assign descriptive names to AS/400 reports and how to index the reports in order to make them searchable by key. Each archive in LaserVault has a Replist file that resides in that archive’s basepath. A LaserVault utility called **Replist Editor** is designed to make it easier for LaserVault administrators to setup and maintain the Replist file.

This document will describe the following:

- A. Report names and descriptions.
- B. Types of keys, conditionals, and modifiers.

## A. Report Naming

### 1. Report Names

When reports are downloaded from an AS/400, one of the parameters in the LVLPROUTQ command or option 11 is called “LaserVault Report Name”. This parameter designates how spool files are given DOS file names when they are transferred from an AS/400 outq to the LaserVault archive’s basepath.

Since \*FILE, the spool file name, can be the same for more than one spool file, a different parameter should be used when designating a report file name for each spool file. In most cases, \*USRDATA is sufficient as it is unique for each spool file. If \*USRDTA is not unique, reports/spool files with the same \*USRDTA will be appended together once they are downloaded into LaserVault. If appending the reports together is NOT the desired outcome, a different attribute, such as \*JOB or \*FORMS, should be used for multiple spool files with the same \*FILE and \*USRDTA.

### 2. Report Descriptions

The report name is not the same as the report description. The report name is the file name that is used by LaserVault to archive the report. The report’s description is seen by users and is typically a more detailed name for the report.

Newly download reports are not viewable until they are indexed. This initial indexing is started by choosing the `Index Reports` option from the `Telnet Operator Menu` or by running option 22 from the operator menu.

By default, after a report or spool file is downloaded and indexed for the first time, its description will be the report’s DOS file name with the word “Report” appended to it. For example, if a spool file’s DOS name is “CDTB”, once downloaded and indexed, its description automatically becomes “CDTB Report”.

In order to make the report description more meaningful to users, the description should be changed to something more explanatory. For example, instead of leaving the report CDTB described as CDTB Report, consider changing its description to “CD Trial Balance Report”.

### 3. Creating and Modifying a Report Description

Report descriptions are stored in the Replist file. The Replist Editor utility, which is located in the `Laservault\Lvus5` program group, can be used to open and maintain the Replist file. After starting Replist Editor, choose `Open Replist` from the `File` menu and browse to the location of the appropriate Replist file.

Another way to start Replist Editor is to start the LVUS management console, click on the `Archives` button, select the archive that contains the Replist file to be changed, and click on `Edit Replist`.

# ReplisT Editor Unleashed

In the ReplisT Editor utility window, the `Reports` window on the bottom right will display the filename(s) of the report(s) that have been downloaded prior to starting ReplisT Editor. The `Commands` window on the bottom left will display the `FILE` command followed by a report name and its default description. The `File` command identifies the downloaded report file by its 1-8 character DOS file name and assigns a default description to the report.

To change the report description, highlight the line that begins with `FILE` and click the `EDIT` button. In the Input field, type a description for the report (up to 47 characters) and click `OK`. The report description will be updated in the `Commands` window immediately.

Another way to change the report description is to highlight a report header or any other string of text in the `Report View` window area, click on the `Misc.` menu and choose `Report Description`. The report description will be updated in the `Commands` window immediately.

**Note:** Once a Report Description has been defined for a report, it should not be changed after archiving has been performed. LaserVault maintains report cross-reference automatically as reports are processed. This cross-referencing allows users to locate a specific report without knowing its associated date/archive through the Report Sequence option. If the Report Description is changed, the Report Sequence will display both the old and the new report descriptions even though although they both refer to the same report.

## B. Keys

### 1. Purpose of Keys

LaserVault gives users the ability to find specific pieces of information in a report or reports without knowing the associated date/archive through the Key Sequence option. Users can also find specific pieces of information in a specific report without knowing where the information is located through Key Lookup. This specific piece of information is known as a key or key value. In order for Key Sequence and Key Lookup to be available, key statements must be defined using ReplisT Editor. A key statement in the ReplisT file is a command that tells the indexing program where to look for a value within each report and how to describe it to the user (i.e. what kind of information it is).

### 2. Types of Keys

#### Simple Keys

Simple keys, `KEY`, look for information on every line of every page of the report. The `KEY` command requires 3 parameters:

- The column where the key value starts.

- The number of characters to include in the index file.

- The key name that serves as a description of the key to the user.

For example:

```
KEY 10 32 Customer Name
FILE CDTB CD Trial Balance
```

The `KEY` command tells the indexing program to look for a value called "Customer Name" that may appear on any line of every page starting in column 10 and that may take up to 32 characters.

#### Advanced Keys

In most cases, using the `KEY` command is not the best way of "telling" the indexing program how to look for and extract values from the report.

# ReplisT Editor Unleashed

In the example above, the indexing program will look on every line of the report for Customer Name information. It will extract whatever information it finds and will store that information as a Customer Name key value in the index file. The index file's purpose is to contain the report's extracted values and remember on what page and line that value appears.

Although the Customer Name does not actually appear on each line, the indexing program will still assume that whatever it finds in column 10 of every line is a Customer Name. The indexing program will store this erroneous information in the index file making the file size larger than necessary and compromising the accuracy of the Key Sequence and Key Lookup processes.

Therefore, keys should be defined carefully so that the indexing program only picks the correct values. Alternative solutions are to exclude unnecessary values by using advanced `KEY` commands or using `KEY` commands together with `Conditional` statements and `Modifiers`.

## Types of Advanced Keys

When creating keys, consider the types of values that the indexing program will be indexing. For values that begin with an alphanumeric character, the `KEY` command should be used. For other types of values, different variations of `KEY` should be used.

### KEYZ

The `KEYZ` command is similar to the `KEY` command except that during the indexing process key values that begin with blanks will be zero-filled instead of being left justified as is normally done. The values inserted into the index file are modified by having zeros added to them.

The syntax of the `KEYZ` command is the same as the `KEY` command.

For example:

**KEYZ 2 6 Item #**

The key value "Item Number" can be up to 6 characters long. Item Numbers that are less than 6 characters long will have "0"s placed in front of them in the index file. When " 10" is indexed, the index file will store "000010" as the key value. In order to retrieve this information, users must insert the zeros in the input field when searching for the key.

### KEYZS

The `KEYZS` command is similar to the `KEY` command except that during the indexing process key values that begin with zeros will be zero-suppressed. The value inserted into the index file is modified by having the extra zeros removed from the beginning of the value.

The syntax of the `KEYZS` command is the same as the `KEY` command.

For example:

**KEYZS 2 6 Customer #**

If the "Customer" value of 10 was printed on a report without zero-suppression, it would appear as "000010", which would normally be stored in the index file as "000010". With the `KEYZS` command, the value is stored as "10" in the index file. In order to retrieve this information, users must remove any leading zeros in the input field when searching for the key. Also, because the leading zeros are removed from the index file, the `Get Next Key Value` operation is affected and 10, not 2, will be displayed immediately after 1.

### KEYLNF

The `KEYLNF` command is similar to the `KEY` command 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 index file as a key value.

## ReplisT Editor Unleashed

The syntax of the `KEYLNF` command is the same as the `KEY` command.

For example:

**KEYLNF 2 25 Customer Name**

The indexing program extracts the last name from the key value by scanning the key value from the end until a blank is encountered before the last name. This will result in incorrect key values for people who have complex last names such as "St. John", which would result in only "John" being moved to the beginning of the key value. One solution to this problem is to remove the space between "St." and "John".

Incorrect key values would also be created if this key statement was used in reports that have a combination of names of people and names of companies because the company names would be reversed in the index file.

`KEYZ`, `KEYZS` and `KEYLNF` scan through every line of every page just like the `KEY` statement does which can make the index file unnecessarily large and potentially inaccurate. A conditional operator or a modifier should be used to limit the indexing program to look for key values only in certain lines where the key values actually do appear.

### KEYP

It is not practical to use the `KEY` command in cases where the value to be indexed appears only once per page. The `KEYP` command tells the indexing program to look for a key value in a fixed place within any page of the report.

The `KEYP` command requires 4 parameters.

For example:

**KEYP 6 10 32 Customer Name**

The `KEYP` command tells the indexing program to look for a value called "Customer Name" that appears only on line 6 of every page starting in column 10 and that may take up to 32 characters. If `KEYP` does not find a value in row 6, it will search the following line(s) until it finds a value in the same column position. This corrects for positioning inconsistencies that can occur during the report generation process.

### KEYPZ, KEYPZS and KEYPLNF

These commands are variations of the `KEYP` command that function the same as the `KEYZ`, `KEYZS` and `KEYLNF` variations of the `KEY` command.

## 3. Compound Key Commands.

Compound indexing commands take values from different places and use them to create a single key. The `KEYSEG/KEYEND` command allows different values from one or more adjacent lines to be combined to create one key. These types of keys allow for values that may appear in more than one column of the report to be defined as a single key.

Compound indexing commands use the `KEYSEG` and `KEYEND` commands. These two statements are only used in conjunction with each other to define a key that consists of multiple segments or values. `KEYSEG` tells the indexing program that this is going to be a segmented key and `KEYEND` signifies the end of a segmented key.

The syntax of the `KEYSEG/KEYEND` command is the same as the `KEY` command.

For example:

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

# ReplisT Editor Unleashed

OR

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

The indexing program will look for Customer # which is located in column 4 for 6 characters long, Invoice # which is located in column 12 and can be up to 7 characters long, and Work Order# located in column 54 and can be up to 7 characters long. The user will look for key value "Cust#/Inv#/WO#".

In order to lookup all 3 values at once using this key, a user would enter all 3 values on the same line starting with Cust# and separate the values with a space. For example, to lookup customer # 456 and invoice # 10025 and work order # 3008 at the same time, the user enters "456 10025 3008" in the input field. LaserVault will display the location of the last portion of the requested information to the user.

Each key segment must appear on the same line or on a following line.

The key description of the first `KEYSEG` is the description that is stored for LaserVault to use. The key description of each successive `KEYSEG` and of the `KEYEND` must include a plus sign (+) or have a plus sign inserted at the beginning. Each `KEYSEG` and `KEYEND` entry counts toward the limitation of 9 keys per report.

All definitions of segmented keys end with `KEYEND`.

## **KEYSEGZ, KEYSEGZS, KEYSEGLNF, KEYSEGP, KEYSEGPZ, KEYSEGPZS, and KEYPLNF**

These commands are variations of the `KEYSEG` command that function the same as the `KEYZ`, `KEYZS` and `KEYLNF` variations of the `KEY` command and the `KEYPZ`, `KEYPZS`, and `KEYPLNF` variations of the `KEYP` command (which tells the indexing program which line to find extract information from).

## **4. Defining Keys with ReplisT Editor**

In the ReplisT Editor utility window, select the filename of the report to be indexed in the `Reports` window on the bottom right.

To add a key, highlight the area to be searched by the indexing program for information in the `Report View` window. Be sure to grab any extra spaces that may contain information on other lines in the report. Click on the `Key` menu and choose the appropriate key type. Enter the key name in the input field. Click `OK`. The key will be added to the `Commands` window immediately.

The `FILE` command will be the last command in the `Commands` window. It tells the indexing program that there are no more keys defined for this report.

## **5. Conditionals**

When a key value always appears on the same line as another character or phrase, a conditional can be used to require that a condition be proven true in order for the key value to be extracted.

### **WHENCOL**

The conditional statement `WHENCOL` looks for a character, word, or phrase that appears in a specific column on the same line as a key value.

The `WHENCOL` conditional requires 2 parameters: the beginning column and string for a comparison. The case of the characters must match that of the string being checked for in the

# ReplisT Editor Unleashed

report. A simple `KEY` or `KEYP` statement should be used after a `WHENCOL` conditional. Use double quotes (") when the string has imbedded blanks within it.

For example:

```
WHENCOL 58 "."  
Key 10 32 Customer Name
```

If there is a decimal point in column 58, only then will the indexing program look for a key value in column 10 that may take up to 32 characters located on the same line as the decimal. This key is named "Customer Number".

If another key value appears on the same line but in a different column, it can use the same conditional by adding another simple `KEY` command that identifies the other key value.

For example:

```
WHENCOL 58 "."  
Key 10 32 Customer Name  
Key 3 6 Customer Number
```

A `WHENCOL` conditional remains in effect until another `WHENCOL`, a `WHENNOTCOL`, or a `CLEARFLAGS` statement is encountered.

*Tip: Characters that are commonly used as conditionals include decimal points in dollar amounts and slashes in dates.*

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

Additional conditionals, such as `ANDCOL`, `ORCOL`, `ANDNOTCOL`, and `ORNOTCOL`, can be added to keys that require more than one `WHENCOL` entry.

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` and `ORCOL` with `WHENNOTCOL` due to the complexity that this adds to the logic. If used incorrectly, all of the possibilities of the given logic may not be properly calculated.

## **6. Modifiers**

In cases when a conditional does not exist for a key value, a modifier may be used. Different types of modifiers should be used depending on the report's layout.

### **AFTERLINE**

`AFTERLINE` is used to specify the line number after which the indexing program can begin checking for key extractions.

`AFTERLINE` requires only one parameter: the line number that denotes that the key value may appear after.

For example:

```
AFTERLINE 6  
KEY 3 6 Customer Number
```

The indexing program will look for the "Customer Number" key value on each line after the first 6 lines.

# ReplisT Editor Unleashed

## ONLYLINE

ONLYLINE is used to tell the indexing program to only extract key values found on the specified line number.

ONLYLINE has no parameters.

For example:

```
ONLYLINE
KEYP 6 3 6 Customer #
```

The indexing program will look for the "Customer Number" key value only on line 6 in column 3 for 6 positions long. If it does not find one, it will not continue looking on the next line.

## NOREPEAT

NOREPEAT 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.

For example:

```
NOREPEAT
KEY 2 6 Customer #
```

## NOREPEATTEXT

NOREPEATTEXT functions like NOREPEAT except that if the same key value continues consecutively across one or more page breaks, an index entry will only be created for the first occurrence and NOT for each page it is on.

For example:

```
NOREPEATTEXT
KEY 2 6 Customer #
```

## CLEARFLAGS

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

For example:

```
WhenCol 66 "/"
Key 20 8 Account Number
ClearFlags
KeyP 9 64 6 Invoice Number
FILE INVST Daily Invoices
```

## 7. Defining Conditionals and Modifiers

In the ReplisT Editor utility window, select the filename of the report to be indexed in the `Reports` window on the bottom right.

To add a conditional, highlight the area to be searched by the indexing program for the conditional in the `Report View` window. Click on the `Conditional` menu and choose the appropriate conditional type. The key will be added to the `Commands` window immediately.

Conditionals and Modifiers should be entered before the key that they refer to, except for `CLEARFLAGS`, which signals the end of the condition.