Suprtool 4.8.02 for HP-UX:

Change Notice

by Robelle Solutions Technology Inc.

Robelle Solutions Technology Inc.
Suite 372, 7360 137 Street
Surrey, BC Canada V3W 1A3

Phone: 604.501.2001
Fax: 604.501.2003

E-mail: solutions@robelle.com
E-mail: support@robelle.com
Web: www.robelle.com

Contents

Introducing Suprtool Version 4.8.02

Overview

Use Suprtool/UX to read, select, and sort data from Oracle, Allbase and Eloquence databases and data files with fixed-length records. Suprtool/UX is designed to be similar to Suprtool for MPE while providing necessary HP-UX features.

Suprlink/UX provides high-speed data-file linking based on a sort key. Use STExport to convert fields in a self-describing input file into an output file that can be imported into different applications.be imported into different applications.

Highlights in Suprtool 4.8.02

- The get command would not get end of file signal properly if the dataset was empty and Set FastRead was on.
- The information about the data loaded in a Table would be lost if the table data being referenced was the second held table and the previous task involved a chain command.

Highlights in Suprtool 4.8.01

- The chain command would not return the correct record when used with Set FastRead On
- The information on the data in a Table would be lost if the table data being referenced was the second held table and the previous task involved a chain command.

Highlights in Suprtool 4.8

- The Clean command in Suprtool would incorrectly upshift lower case alpha characters.
- The $edit function will now work when nested within other string functions
- The List command on HPUX now has the RECLEN parameter.
- Suprtool now has the command Set Oracle ZeroNull On or Off which optionally turns null values into zeros.

Highlights in Suprtool 4.7.12

- Suprtool for HP-UX now can read Eloquence datasets up to five times faster than the default DBGET serial reads.
- Eloquence routines are now dynamically loaded, this allows new features and bug fixes to be available as soon as the new version of Eloquence is loaded.
- The $subtotal command has been re-written to use less resources and some bugs have been fixed.

Highlights in Suprtool 4.7.11

- Suprtool now has a $findclean function to identify records with specific characters in it
- Dbedit for HP-UX now works on HP Eloquence databases.
- The List command now has new options for listing to a file on HP-UX.
- The $subtotal function would not work if the previous task used the Dup None Keys feature.
- The $total function would appear to total data incorrectly when sorting on the field that was being totalled.
- Variables that resolved to all spaces for the entire command line would not work.

Highlights in Suprtool 4.7.10

- Suprtool now allows up to 255 $split functions per task.
- Suprtool now has a $edit function for formatting data.
- Suprtool's Open command can now connect to a remote Oracle database.

- Numrecs 100% would come up with the wrong output file size when reading very large files.
- The new $split function would put random characters at the point where the split would occur in some cases.
- The new $split function would incorrectly report an error in a second task with multiple $split operations.
- The new $number function did not handle numbers that consisted of only a decimal place followed by any number of zeroes and a number, as in .01 thru .09.
- Suprlink would abort if the Join file was empty.
- Suprlink would hold the Join file open after the task was completed.
- Suprtool would total incorrectly when using a $subtotal function.
- The form command would display the percentage full as a very large strange number if the capacity and the entries were both zero for Eloquence databases.
- Update ciupdate would not be effective for Eloquence databases in the case where Ciupdate was allowed.
- We have worked around an Oracle patch issue that stopped Suprtool from being able to connect to databases with the Open command.

Highlights in Suprtool 4.7.02

- The $number function would report and incorrect error message in some cases.
- An update of a critical item would fail with an incorrect status.

Highlights in Suprtool 4.7.01

- The base command would fail with a dblogon error message if security on an Eloquence database was setup a certain way.

Highlights in Suprtool 4.7

- Suprtool now has a $Number function which will allow Suprtool to read a freeform ascii number with signs, decimal places and currency symbol as a display field.
- Suprtool now has a $SubTotal function.
- Suprtool can now split byte strings into multiple fields via the $split function.

- Suprlink can now do many-to-many links via the Join command.
- Suprtool and STExport now support features to Clean your data.
- STExport now supports an Escape command, which will escape out certain characters when used with certain import programs.
- Suprtool now supports a $counter function which will increment a field by one for every record selected.
- Suprtool now supports a $total function which will keep a running total for a specified field.
- Suprtool, STExport and Suprlink now support environment variables.
- Suprtool has been enhanced to support Eloquence 7.0 features.
- The Base and Put commands have been changed to support new features in Eloquence 7.0.
- Suprtool now supports the expanded database limits in Eloquence 7.0.
- Suprlink and STExport now report the number of output records in a manner similar to Suprtool.
- The Table command now supports filenames up to 80 characters.
- The Table command truncated filenames at the limit of 36 characters.
- The number of defines allowed in Suprtool has been increased to 768.
- Suprtool would report an incorrect record number when encountering an Illegal ascii digit, if the Suprtool task involved the Duplicate command.
- Suprtool would incorrectly report an error on a second set tablesize command for a given task.
- STExport would in some cases attempt to format data in XML and HTML format at the same time.
- Extract from a table would incorrectly report an error in some cases.

Known Problems

There are two known issues associated with the Oracle interface and specifically with the Open command.

In order to fix a problem introduced by a patch to Oracle 9, we had to change to a new Oracle call interface call. You can invoke this new call in one of two methods, the first is to use the following Set command:

```
>Set Oracle OpenFix On
```

The second method is to specify the username/password in the following manner:

```
>open oracle scott/tiger
```

When using this type of syntax and the Open command is preceded by a system command, such as echo, the Open command will fail with a core dump. The work around is to remove any system commands prior to the open command.

In order to connect to the remote database using Suprtool, you need to specify the connection parameters as follows:

```
>open oracle username/password@machine
```

When using this syntax, Suprtool uses a different method for connecting to a database, which for some reason fails on the second and fourth connections. While we normally would not release software with this kind of defect, we have done so for two reasons:

1) We cannot find any reason in our code for the problem and the failure appears to occur inside the Oracle call.

2) Some customers that need the functionality to connect to a remote machine are willing to live with the problem.

Suprtool 4.6 and Suprtool 4.7 by default would forcefully return zeroes for fields which were considered null. It is important to note that Suprtool would return zeroes for Null fields, it is just Suprtool 4.6 and 4.7 forced this as some null fields for a customer were corrupt. We have made this optional with Set Oracle NullZero with the default being off.

Suprtool by default no longer forces zeroes for null numeric fields as Suprtool incorrectly zeroes out a field if the Select statement contains the to_char function and has the field has null values.

This should be fixed in a future version of Suprtool.

If you have any questions or concerns or feedback on this or any other issue, please feel free to e-mail me at: neil@robelle.com

Compatibility

Suprtool/UX is compatible with HP-UX 9.0, all versions of HP-UX 10.x, all versions of HP-UX 11.x, as well as Oracle version 7.1.3.2.0. On HP-UX 10.2x, Suprtool/UX creates all of its temporary and scratch files in /var/tmp, unless you have overridden the temporary directory with the |4TMPDIR| environment variable.

Suprtool for HP-UX typically comes with two versions in two different directories on your tape. The version of Suprtool in /opt/robelle is compatible with HP-UX 10.20 and later. The version of Suprtool in /usr/robelle is compatible with versions earlier than HP-UX 10.20.

Previously Suprtool would truncate a Table filename at 36 characters if the filename was greater than 36 characters and therefore would open a file if a file existed at the previous 36 character limit. The Table filename has been increased to 80 characters. If the Table filename exceeds 80 characters Suprtool will print an error.

CPU Serial Number (uname)

This program runs only on CPUs whose serial numbers have been encoded (the "uname" on HP-UX). If it fails to run and you get an "invalid HPSUSAN" error message, contact Robelle for assistance, via support@robelle.com or the support number at 1-800-453-8970.

Documentation

The user manual contains the full description of all the Suprtool suite of products including Dbedit, Suprlink, STExport, Dbedit and Suprtool2, as well as usage tips and commands for each. The manuals are up-to-date with all the latest changes. To see only the changes in the latest version, see the "What's New" section of the manual, or see the change notice.

You can download our manuals and Change Notices in PDF format or HTMLHelp format (.CHM) and even order printed (hardcopy) manuals from our web site at:

http://www.robelle.com/library/manuals/.

Installation

Overview

The following instructions describe the installation process of a new Suprtool release. The new version overwrites an existing version of Suprtool on your HP-UX system.

Installation Instructions

There are typically two main types of installations. The first and most often utilized is the Download instructions. You can find the HP-UX download install instructions here:

http://www.robelle.com/downloads/install-sxprod.html

Tape installation instructions can be found here:

http://www.robelle.com/support/install/tape/sxprod.html

Installation Assistance

If you have any questions or run into any problems, please call us. Technical support is available on weekdays from 8 a.m. to 4 p.m. Pacific time at 1.800.453.8970. Technical support can also be obtained via e-mail at: support@robelle.com If your new version of software will not run, you can page someone from technical support by calling the 1.800 number, or you can typically easily run extend with the disaster option to tide you over until business hours. Instructions for this are available at:

http://www.robelle.com/disaster/

Enhancements in Version 4.8

Introduction

Every year we provide Suprtool users with new features. The following section describes the new enhancements to Suprtool since Suprtool 4.7.

Set Oracle ZeroNull

Suprtool for HP-UX has a new option to turn on changing null fields to zeroes. Set Oracle ZeroNull On will change any fields that are "null" (in the SQL sense of the word), will become zeroes for the appropriate number type.

In order to turn this feature on for all accesses you can put the command:

```
Set Oracle ZeroNull On
```

into the file /opt/robelle/suprmgr.

Previous to version 4.6 in Suprtool used to return nulls, but Suprtool 4.6 and Suprtool 4.7 would return zeroes. We have decided to make this optional and make the default to return Nulls, due to problems when using the to_char function in the select statement. See the Compatibility section for a more detailed explanation.

Reclen on List command

Suprtool for HP-UX can output data from the List command to a discfile. Although the concept of record size is not the same on HP-UX as it is on MPE it is still important in some areas within Suprtool.

In this case the RECLEN parm merely tells the List command where to fold the lines. The Reclen parm can be a value from 56 to 256 and defaults to 80 bytes.

```
>List FILE myreport RECLEN 132
```

Enhancements in Version 4.7.12

Introduction

Every year we provide Suprtool users with new features. The following section describes the new enhancements to Suprtool since Suprtool 4.7.

Set FastRead On

Suprtool for HP-UX by default calls dbget to do serial reads, now with Suprtool you can utilize faster reads with the Set FastRead On command. This command invokes more efficient large reads. Testing has shown that the CPU time can be improved by anywhere from two to five times and Wall time has improved anywhere from two to six times faster. In order to turn this feature on for all accesses you can put the command:

```
  Set FastRead On
```

into the file /opt/robelle/suprmgr. This means that Suprtool will use the faster reads for all runs of Suprtool.

Dynamic Loading

Suprtool for HP-UX now attempts to dynamically load the Eloquence routines. Suprtool requires two Eloquence libraries, namely: libimage3k.sl and libeqdb.sl. What Suprtool now does is look for these libraries in two separate ways.

First Suprtool looks for libeqdb.sl and libimage3k.sl in any of the directories named in the SHLIB_PATH. For example to insure that Suprtool resolves the library loads you can set the SHLIB_PATH system wide in your /etc/profile file in the following manner:

```
# add SHLIB_PATH for Eloquence library search
    export SHLIB_PATH=/opt/eloquence6/lib/pa11_32
```

The line proceeding the export command is a comment line and does not need to be in the file, but is just a reference to indicate what it is used for. If you do not have the SHLIB_PATH variable set to a value where libeqdb.sl and libimage3k.sl can be found, Suprtool will then try to load libimage3k.sl in the directory /opt/eloquence6/lib/pa11_32 and libeqdb.sl from the same directory. If the libraries still fail to be loaded then Suprtool will print two warnings, however, it will still continue to function, just any of the Eloquence features will fail when called:

```
SUPRTOOL/UX/Copyright Robelle Solutions Technology Inc. 1981-2004.
(Version 4.7.12 Internal)  MON, MAR 08, 2004,  7:34 AM  Type H for
help.
Warning: Could not load Eloquence image library
Warning: Could not load Eloquence scan library.
```

Enhancements in Version 4.7.11

$FindClean Function

We recently added the $Clean function to primarily clean "bad" characters in text fields. This has been extremely popular enhancement but many wanted to do investigative work and try to figure out what records had these bad characters, to hopefully find out where the "bad" data was coming from. For this reason we have created the $FindClean function. $FindClean will return true if it finds a character defined using the Clean command.

```
>in cleansd
>clean "^9","^10"
>if $findclean(nonprint)
>list
>xeq
```

The above task will list the record if the field nonprint has a Tab (Decimal 9) or a Line Feed (Decimal 10) anywhere in the field. You can Find and clean the "bad" characters from a field at the same time:

```
>in cleansd
>clean "^9","^10"
>if $findclean(nonprint)
>extract nonprint=$clean(nonprint)
>list
>xeq
```

Dbedit Module

The Dbedit Module has been added to Suprtool for HP-UX. It works with Eloquence databases and with Oracle databases with the IMAXSoft version of Suprtool.

List Command

The List command has been enhanced with a new File keyword that allows output to be directed to a file. The List command also has a new option to Append to an existing file. The File option takes the next parameter as being the filename:

```
>in test/file1sd
>list stan file myslist
>xeq
```

If the file myslist exists it will be over-written, unless you specify the Append option. If you specify the append option the new report will be added to the file.

So if you want to incorporate multiple reports you just need to do the following:

```
>in test/file1sd
>list stan file myslist
>xeq
>in test/file2sd
>list stan file myslist append
>xeq
```

Enhancements in Version 4.7.10

$Split Function

Suprtool now allows up to 255 $split functions per task. The previous limit was 16, and the limit has been changed to assist in reading data from "PRN" files.

$Edit Function

Suprtool can format fields using edit-mask features similar to edit-mask features of Cobol. Suprtool employs two distinct types of edit-masks: one for byte type fields and the other for numeric fields.

The type of mask utilized depends on the source type of the field. If the source field is numeric, then the numeric edit-mask logic is applied, if the source field is byte type, then the byte edit-mask logic and characters apply.

The target field must always be a byte type field.

Placeholders and Format Characters

An edit-mask consists of "placeholder" characters, such as "9" for a numeric column, and "format" characters, such as "." for the decimal place. Sometimes an edit-mask character acts as both a placeholder and a format character, such as the "$" in floating dollar signs.

Byte-Type Formatting

For Byte type fields there are two placeholder characters. These are:

X ~ place the data in the matching column for the X in the edit-mask

Z ~ place the data in the matching column unless the data is a zero; if the data is a zero, then replace with a space

The format characters are as follows:

B (space) / (slash) , (comma) . (period) + (plus) - (minus) * (asterisk)

and a Space. Please note that you can denote a space using two methods, either by putting a "B" in the mask or a space itself. For example, suppose you have data that is in ccyymmdd format in an X8 field. Here is how you would use a "xxxx/xx/xx" mask to format the data:

```
>in mydate
>form
   File: MYDATE.TEST.NEIL (SD Version B.00.00)
      Entry:             Offset
         A       X8      1  <CCYYMMDD>
   Limit: 10000  EOF: 2  Entry Length: 8
>def formatdate,1,10
>ext formatdate=$edit(a,"xxxx/xx/xx")
>list
>xeq
>IN MYDATE.NEIL.GREEN (0) >;OUT $NULL (0)
FORMATDATE   = 2003/09/24

>IN MYDATE.NEIL.GREEN (1) >;OUT $NULL (1)
FORMATDATE   = 2003/09/24
```

As you see in the example above, the placeholder character is the "x" and the "/" is the format character. You insert a space either by specifying a "B" or by putting an actual Space character in the edit-mask. An example of inserting a space might be the formatting of Canadian postal codes (e.g., V3R 7K1):

```
>in postal
>form
   File: POSTAL.NEIL.GREEN
      Entry:                    Offset
         POSTAL-CODE          X6      1
   Limit: 10000  EOF: 2  Entry Length: 6
>def post1,1,7,byte
>def post2,1,7,byte
>ext post1=$edit(postal-code,"xxx xxx")
>ext post2=$edit(postal-code,"xxxbxxx")
>list
>xeq

>IN POSTAL.NEIL.GREEN (0) >OUT $NULL (0)
POST1     = L2H 1L2     POST2     = L2H 1L2

>IN POSTAL.NEIL.GREEN (1) >OUT $NULL (1)
POST1     = L2H 1L2     POST2     = L2H 1L2
```

Z-placeholder for byte-fields

The Z-placeholder character works differently for byte-fields than for numeric fields. For byte type fields, if the Z placeholder and the corresponding data is "0", then the zero is suppressed, regardless of the position. This is primarily for suppression of zeroes in byte type date fields:

```
ext a=$edit(date-field,"xxxx/zx/zx")
```

The above edit mask would then edit a byte type date of 20031005, to be:

```
2003/10/ 5
```

Overflow and limits

An edit mask is limited to 32 characters in total for both numeric and byte type fields. If data overflows the edit-mask, by default Suprtool will fill that field with asterisks. There is an option to have Suprtool stop when it encounters a formatting overflow:

```
>set editstoperror on
```

will force Suprtool to stop if there is data left over after applying the edit-mask. With byte-type fields, leading spaces do not cause overflow. Therefore if your data consists of:

```
"    L2H1L2"
```

and your edit mask is:

```
"xxxBxxx"
```

It is not an overflow since there are only spaces to the left of the "L". If the data was:

```
"   JL2H1L2"
```

an overflow exception would occur.

Numeric field edit-masks

Our edit-masks for numeric fields are patterned after those in COBOL. We provide four placeholder characters, each with a slightly different effect:

"9" - insert a digit from 0 to 9 in this position

"$" - if you specify more than one dollar sign, you get a floating dollar sign. This means that there can be as many numeric positions as there are dollar signs, but if some positions are not needed because the value is small, the $ floats to the right next to the first digit and the preceding positions are blank.

"*" - if there are enough digits in the value, the * position is replaced by a numeric digit; if not, an asterisk is printed. Leading asterisks are often used for check writing, so that no one can insert a different value.

"z" - insert a numeric digit at this position; if the rest of the data to the left is a zero then a space will be placed at this position. For example:

```
>ext a=$edit(int-field,"$$,$$$.99-")
>ext b=$edit(int-field,"99,999.99-")
>ext c=$edit(int-field,"cr99999.99")
>ext d=$edit(int-field,"-$9999.99")
>ext e=$edit(int-field,"**,***.99+")
>ext f=$edit(int-field,"zz,zzz.99+")
>list
>xeq
>IN FILE1SD.NEIL.GREEN (0) >OUT $NULL (0)
A      =    $11.11-     B   = 00,011.11-
C      = CR00011.11     D   =  -$0011.11
E      = ****11.11-     F   =     11.11-

>IN FILE1SD.NEIL.GREEN (1) >OUT $NULL (1)
A      =    $22.22-     B   = 00,022.22-
C      = CR00022.22     D   =  -$0022.22
E      = ****22.22-     F   =     22.22-
```

Signs

As shown in the example above, there are also numerous format characters for numeric edits, including four ways to specify the sign. You can specify a sign, with +, -, or the typical accounting specification of "CR" and "DB". You will note in the example above that the "cr" in the mask was up-shifted to be "CR". This is because the entire mask is up-shifted as the mask is being parsed.

You can specify more than one sign in a numeric field edit, although Suprtool will give you a warning that having two sign edit-mask characters does not really make sense. Cobol gives a Questionable warning when compiling an edit-mask with two sign characters. Suprtool, will apply the sign in both places.

Keep in mind that most data has three states:

1) Postive
2) Negative
3) Neutral

Any neutral data will not display the sign. If you specify a "+" sign in the edit-mask and the data is negative, it will of course display a "-" sign.

Decimal Places

For numeric-type edits, Suprtool attempts to adjust the data according to the number of decimal places in the edit-mask, when compared to the number of decimal places defined in the field.

For example if the data field has one decimal place, and the edit mask has two decimal places, then the data is adjusted:

Data and Edit mask:

```
102.3   ZZZZ.99
```

will result in the final data being:

```
  102.30
```

Similarly, if the data has three decimal places and the edit-mask only has two, then the data will be rounded appropriately with the same rules as outlined in the $number function.

You can specify more than one decimal place in an edit-mask. However, Suprtool will print a warning and it will utilize the right-most decimal place for data alignment. The decimal place character is defined by a set command:

```
   >set decimalsymbol "."
```

If you define another character as the decimal symbol, Suprtool will use that character as the point to align the decimals. If you define a decimal symbol that is not an allowed edit-mask character with Set Decimalsymbol, Suprtool will assume that the field has zero decimal places and adjust the data accordingly.

Currency and Dollar signs

Suprtool edit-masks support both fixed and floating dollar signs. Logic for floating dollar-signs will be invoked if more than two dollar signs are defined in the edit-mask.

A floating-dollar edit mask attempts to put the dollar sign at the left most position of the significant data. For example if you have the following data and edit mask:

```
0001234.54 $$$$$$.$$
```

the data would end up as:

```
   $1234.54
```

Suprtool will not however, put the dollar sign to the right of the decimal place. If you had the same edit mask and the data was, .09, the data would end up being formatted as:

```
       $.09
```

Similarily, the $edit function will attempt to place the dollar sign correctly in most cases. For example Suprtool will not format data in the form of:

```
   $,123.50
```

Suprtool, does attempt to fixup these cases and would format the data in the following manner:

```
    $123.50
```

Overflow and floating dollar

If the number of digits in the data is equal to the number of placeholder dollar signs, then the dollar sign is dropped and not added to the edited field.

```
12345.50 $$$$$.99
```

would result in:

```
12345.50
```

Set CurrencySymbol

If Set CurrencySymbol is not equal to "$", then after the formatting has been applied, whatever symbol(s) are defined within the set command, are used to replace the "$" symbol in the data. For example, if you have the Currency symbol set as "CDN".

```
>set currencysymbol "CDN"
```

Suprtool will replace the "$" after the edit-mask has been applied with CDN, provided there is room to the left of the dollar-sign.It is recommended that if you are using multiple characters for the dollar symbol that you leave enough characters to the left of the symbol.

For example if the CurrencySymbol is defined as CDN, then you should leave two spaces to the left of a fixed dollar sign definition. If there is not enough room, to put in the currency symbol, then the dollar symbol is blank.

Overflow and limits

An edit mask is limited to 32 characters in total for both numeric and byte type fields. If data overflows the edit-mask, by default Suprtool will fill that field with asterisks. There is an option to have Suprtool stop when it encounters a formatting overflow:

```
>set editstoperror on
```

will force Suprtool to stop if there is data left over to place when applying the edit-mask. With numeric-type fields, leading zeroes do not cause overflow.

Oracle Open

The Open command has been enhanced to allow connections to a remote Oracle database. In order to invoke this feature the username/password@machine, need to be specified on a single line.

```
>open oracle suprtest/suprpass@remote
```

There are some issues with using this syntax, for more information please read the section entitled Known Problems.

Enhancements in Version 4.7

Eloquence 7

Suprtool has been enhanced to work with Eloquence 7. It takes advantage of many new features and supports the new limits and data types.

Base Command

Suprtool's Base command has been enhanced to allow the new syntax supported in Eloquence 7. Eloquence now allows the server name and service to be specified in dbopen. To support this new syntax Suprtool's Base command has been changed to allow the server name, service and database name be specified. Suprtool uses the same syntax as Eloquence whereby the database name consists of the following elements:

```
[[host][:service]/]database
```

Examples of using this syntax within Suprtool using the sample database that Eloquence provides.:

```
base sample,5
base :eloqdb/sample,5
base hostname.robelle.com:eloqdb/sample,5
```

Put Command

The Put command has always supported for a database name to be specified. Since the Base command allows the new syntax for Eloquence version 7 support, the Put command also allows this new syntax.

```
put dataset,sample
put dataset,:eloqdb/sample
put dataset,hostname.robelle.com:eloqdb/sample
```

Expanded Database Limits

Eloquence version 7 now has new database limits that follow or extend the current Turbo IMAGE limits.

```
Data Items per database        2048
Data Sets per database         500
Paths per dataset detail       16
Paths per dataset master       64
```

Variable Substitution

Suprtool now supports environment variable substitution. To use this enhancement you must do a:

```
>set varsub on
```

Due to how HP-UX processes work with environment variables any variables must be exported prior to running Suprtool, STExport or Suprlink. All of these programs support HP-UX variable Substitution.

```
export tablefile='abcdefghijklmnopqrstuvwxyzabcd'
export infile='file1sd'
./suprtool -oc << \!EOD
 set varsub on
 in $infile
 table mytable,char-field,file, &
    /users/robdev/suprtool/test/$tablefile
 if $lookup(mytable,char-field)
 out file05,link
 exit
 !EOD
```

Suprtool examines a command line and looks for variables denoted by the "$" sign. Since Suprtool has some functions that begin with a $-sign, these will take precedence regardless of the value set in the variable of the same name.

Tables on HP-UX Improvements

We have re-written the underlying mechanisms for the Table and $Lookup functions in Suprtool for HP-UX, to allow more data in the table and have similar functionality to the Table command on MPE. This includes Larger tables and Extract from a table functionality described below.

Table Sizes

On HP-UX you can control the size of a table with the Set Limits TableSize command. By default an individual Table will be 50 Megabytes in size and you can have up to 10 tables. The Global limit for all tables is up to 500 Megabytes. You can control the size of a given table with the command:

```
>Set Limits TableSize n
```

If you enter the command Set Limits TableSize 100 and the next table command that you build will have a limit of 100 Megabytes. Previously the Limits on Tables were 15Mb in total and defaulted to 1Mb in size.

Extract from a Table

Suprtool now has the ability to load data into a table via the Table command, and extract that data out of the table using the Extract command. The Table command now allows for data to be loaded along with matching key values.

```
>table table-name,key-field,file,filename,&
 data(field1,field2,...)
```

An example of loading two data fields called cost and desc along with the key field of part into a table would be:

```
>table partab,part,file,partin,data(cost,desc)
```

You can specify up to 20 data fields as long as the total size of the key fields and data does not exceed 256 bytes. The Table file must be Self-Describing (Link) in order to use the data option. When loading data into a table, Suprtool will eliminate the duplicate entries based on the key value, so the associated data values may not be loaded into the table.

The Extract command can utilize the $lookup function to return data. The syntax for the $lookup function would look as follows:

```
>extract target = $lookup(table-name,key-field,data-field)
```

The Table name, key-field and data-field are all defined by the Table command, which must be input before the Extract command.

A classic example: your boss comes to you with a list of new prices and descriptions for certain parts for your Part-Master dataset.

The basic steps to do this are to load the new prices and descriptions into a Table, index by the product number (prodno), then Extract the price field from each record and replace it with a $lookup on the table.

Here is the Suprtool code:

```
>table newprices,prodno,file,bosslist,data(price,desc)
>get part-master
>if $lookup(newprices,prodno)
>update
>extract price = $lookup(newprices,prodno,price)
>extract desc = $lookup(newprices,prodno,desc)
>xeq
```

We do the If $lookup to select only the parts which have new prices, then do Extract with $lookup to replace the existing price with a new one. The Update command forces a database update on each selected record and must come before the Extract command.

If you did not specify the If $lookup, then records that did not qualify under the $lookup function in the extract field, will result in zeroes for any numeric field and spaces for any byte type fields.

$Number Function

Suprtool now has the ability to accept free-form "numbers" as display data types. This means numbers in the form:

```
1234.45-
-12345
-123.2134
 12343
$123.45
```

can now be accepted and converted to any other numeric data type. Consider the following data:

```
  Item-number    New-Price
   12345          +123.45
   34563          + 27.5
   21312          +  1.545
```

Suprtool can now read and convert the data in New-Price using the number function. Let's say we want New-Price to be a double integer but currently occupies eight bytes starting in position six. Here is the task you would use to convert the New-Price free-format number into a double integer.

```
  >in mynums
  >def item-number,1,5,byte
  >def new-price-ascii,6,8,display
  >def new-price,1,4,double
  >item new-price-ascii,dec,2
  >item new-price,dec,2
  >ext item-number
  >ext new-price=$number(new-price-ascii)
  >out somefile,link
  >xeq
```

The $number function will takes the free-format number and makes it a valid display number. It will determine the decimal, sign and add leading zeroes. It will round the number to the defined number of decimal places.

In the case of the 1.545 number, Suprtool will round the value to be 1.55, since the given number of decimal places is two and the following value is five or greater. If you have a whole number such as 54, with no decimal point the value becomes 54.00.

Suprtool will not accept data that has:

```
   More than one sign.
   More than one decimal place.
   Spaces in between numbers.
   Signs that are in between numbers.
   Characters that are not over punch characters.
   Fields that when edited do not fit in the defined space for the
display field.
```

You can control the character that defines the currency, thousand and decimal symbol for other currencies and formats using the following commands:

```
   >set decimalsymbol "."
   >set thousandsymbol ","
   >set currencysymbol "$"
```

Suprtool in the above case will strip the currency and thousand symbols and use the decimal symbol to determine the number of decimal places. You can set these characters to any values you want but the defaults for each are used in the above set commands. The decimal and thousand symbols are only single characters. The currency symbol allows for four characters.

Suprlink Join Command

Suprlink can now join files together that have multiple key records in each file, what has been come to be known as a many-to-many link. Suprlink has traditionally been able to link an Input file with many records with the same key to a Link file that has a single record with the same key value.

The Join command, will now link two files with many key records in both the input file and the "Linking" or join file. The syntax of the Join command is exactly the same as the Link command so a sample task would look as follows:

```
+input ordhist
+join  orders
+output custord
+xeq
```

The above task will link multiple records of the file ordhist, to the multiple records of the file in orders. This assumes that the files are sorted by a common key. In SQL terms this is known as an Inner Join. An Outer Join, one where the keys do not necessarily have a match can be achieved by adding the optional keyword to the Join command:

```
+input ordhist
+join  orders optional
+output joined
+xeq
```

In SQL parlance, once again you can achieve both a Left Outer Join and Right Outer Join by reversing the order of the files, between the input and the join commands.

To give you an example of how the Join operation would work consider the following data. First we have an inventory file with multiple records for the same product-no. This data is stored in the file dinv:

```
50512001  {Rest of data}
50512001  {Rest of data}
50512003  {Rest of data}
```

The next file will have sales records, once again with multiple key values, this data is stored in the file dsales:

```
50512001  {Rest of data}
50512001  {Rest of data}
```

If you did the following task assuming both files are sorted by the product-no:

```
+in dinv
+join dsales
+out invsales
+xeq
```

The resulting file would have four records, with the multiple matching dinv and dsales records. The record layout would have the dinv information first followed by the dsales information. If you add the optional keyword on the join command the resulting file would have 5 records. The matching 4 records from dinv and dsales as well as the dinv record that did not match with the numeric fields set to zero and the byte fields set to spaces.

Only one Join operation is allowed per task.

By default, Suprlink will join files based on the primary sorted key in the self-describing file. You can specify a secondary key for the files to be joined on in a similar manner to how the Link command did:

```
+in orders
+join dsales by order-no product-no
+out ordsales
+xeq
```

$Split Function

Suprtool can extract portions of a byte field based on the occurrence of certain characters.

Consider the following data:

```
Armstrong/ Neil/ Patrick
Green/ Bob/ Miller
Fritshaw/ Elizabeth/
Edwards/ Janine/
Armstrong/Arthur/Derek
```

The $split function can extract each token into separate fields. The syntax for the $split function is:

```
   $split(Field,Start Character,Occurrence,End Character,Occurrence)
```

The following task will split the data in the wholename field into three separate fields. This task assumes that the file namefile is a self-describing file with a field called wholename.

```
  >in namefile
  >define lastname,1,30
  >define firstname,1,20
  >define middlename,1,20
  >extract lastname = $split(wholename,first,"/")
  >extract firstname=$trim($split(wholename,"/","/"))
  >extract middlename=$trim($split(wholename,"/",2," ",2))
  >out names,link
  >xeq
```

The first extract statement tells Suprtool extract the bytes from the field wholename, starting at the beginning (first keyword), and stopping at the "/" character. The second extract statement, tells Suprtool to extract the bytes between the first occurrence of the "/" character to the next occurrence of the "/" character, and then that string is trimmed of spaces as it is nested within the $trim function.

The third and final extract statement tells Suprtool to extract the bytes beginning with the second occurrence of the "/" character to the second occurrence of the space character.

If the target field is not long enough to hold the data Suprtool will abort with an error. You can easily prevent this from happening on blank fields by nesting the $split statement within a $trim or $rtrim function.

The $split function also has a Last keyword, whereby you can split the field from a given occurrence of a character to the end of the field. So in the given example from above the extracting out of the middlename could be coded as such:

```
  >extract middlename=$trim($split(wholename,"/",2,last))
```

The above means to extract out all the data from the second occurrence of the "/", to the end of the field and trim all spaces.

$SubTotal Function

Suprtool has the ability to keep a running subtotal for any numeric field based on a given sort key. The target data must be a packed field with 28 digits, in order to help avoid overflow issues.

A sample use of the $subtotal function could be:

```
>def mytotal,1,14,packed
>get orders
>sort order-number
>ext order-number
>ext part-number
>ext description
>ext sales-amount
>ext mytotal = $subtotal(sales-amount,order-number)
>out sales,link
>xeq
```

This would result in a file containing a running subtotal in the field mytotal for a given order-number. You could then generate a simple report with the simple Suprtool commands:

```
>in sales
>list standard
>xeq
```

The basic syntax for the $subtotal function in the extract command is:

```
extract targetfield = $subtotal(field,sort-field)
```

You must specify the sort command before referencing the sort-field in the $subtotal function. You can subtotal up to ten fields per pass and the $subtotal function is also available in the if command, however, is of limited use.

Cleaning your Data

In this day and age of migrations we were looking at issues that customers have run into when importing data into new databases. What came from this investigation where ways to clean up your data in any given byte type field.

We have added two methods to clean your data, you can use Suprtool to clean an individual byte type field, or STExport to clean all of the byte-type fields for a given file that you are exporting.

Suprtool

Sometimes un-printable or extraneous characters get stored in files or databases that have no business being there. This may be some tab characters in an address field or perhaps and embedded carriage return or line-feed. Suprtool now supports the $Clean function which will replace individual characters for a given byte field.

There are three things that Suprtool needs to know in order to "clean" a field. Suprtool needs to know which characters it needs to clean, what character it needs to change the "bad" characters to, and also what field does it need to clean.

Defining a Clean Character

The Clean command is used to tell Suprtool what characters it needs to look for in a given byte type field. For example:

```
   clean "^9","^10","."
```

will tell Suprtool to replace the tab character (Decimal 9), Line Feed (Decimal 10), and a period to whatever the CleanChar character is set to. The Clean command takes both, decimal notation and the character itself, however, it is probably most convenient to use the decimal notation for the characters that you wish to clean, since most are un-printable. The decimal notation is indicated by the "^" character.

The Clean command also has a special keyword which defines special characters Decimal 0 thru Decimal 31 via the command:

```
Clean special
```

You can also specify a range or characters by using the following syntax:

```
Clean "^0:^31","^240:^255"
```

Setting the Clean Character

By default, Suprtool will replace any of the characters specified in the clean command with a space. You can specify what character to use to replace any of the characters that qualify with the following set command:

```
>set CleanChar "."
```

This will set the character to replace any of the qualifying "to be cleaned" characters to be a period.

Cleaning a Field

You call the clean function, the same way you normally use other functions available to if and extract. For example:

```
ext address1=$clean(address1)
```

shows how to clean the field address1. You do not necessarily need to have the target field be the same as the source field.

```
def new-address,1,30
ext new-address=$clean(address1)
```

Cleaning your data

An example of how easy it would be to clean your database of certain "bad" characters in byte-type fields would be as follows:

```
>base mydb,1,;
>get customer
>clean "^9","^10","^0","^7"
>set cleanchar " "
>update
>ext address(1) = $clean(address(1))
>ext address(2) = $clean(address(2))
>ext address(3) = $clean(address(3))
>xeq
```

The above task will look at the three instances of address and replace the tab, linefeed, null and bell characters with a space.

STExport

This same feature has been added to STExport, except that STExport will automatically clean all the byte type fields for a given SD file. The commands are very similar, except STExport just needs to know what the replace character should be and what characters it needs to look for.

```
$ in mysdfile
$clean "^9","^10","^0","^7"
$set cleanchar " "
$out myexport
$xeq
```

Since the Cleanchar is by default set to space, the above task could simply be:

```
$in mysdfile
$clean "^9","^10","^0","^7"
$out myexport
$xeq
```

Outcount and Fullcount

Suprlink and STExport on HP-UX now have a method for reporting how many records have been output. In Suprtool, we wrote the number of records to a file called .stoutcount. In Suprlink and STExport we used the files, .sloutcount and .sxoutcount respectively.

For Suprlink:

```
&#!/bin/sh
&#
suprlink -oc << !EOD
+in orders
+join ordhist
+out ordcomb
exit
!EOD
if [ `cat .sloutcount` -ge 10 ]; then
   echo "More than 10 records found"
fi
```

For STExport:

```
&#!/bin/sh
&#
stexport -oc << !EOD
$in orders
$heading fieldnames
$out ordprn
exit
!EOD
if [ `cat .sxoutcount` -ge 11 ]; then
   echo "More than 10 records found"
fi
```

STExport Escape Command

Many SQL importers allow you to add an escape character in front of characters that may mean something else to the import program. For example if the import program thinks that the delimiter character is a comma, the importer may treat a comma in an address field as an indication to move to the next field, which will throw of the import.

Some import programs, will treat the next character as data as opposed to a delimeter if the character is preceded by an escape character, such as a slash. Thus when the field is analyzed by STExport the data that originally started as:

```
"Niagara Falls,Ontario, Canada"
```

would be transformed to be:

```
"Niagara Falls/,Ontario/, Canada"
```

This function will not work on fixed columns and can be invoked with the escape command:

```
escape delimeter quote eol "/"
```

The above command will take the defined delimeter, quote and Eol and escape with a "/", if found in any byte type field.

Table Filename

Previously the permitted length for the filename for the Table command was 36 characters. This has been increased to 80 characters.

$Total Function

Suprtool now has the ability to keep a running total for any numeric field. The target data must be a packed field with 28 digits, in order to help avoid overflow issues. A sample use of the total function could be:

```
>def mytotal,1,14,packed
>get orders
>ext mytotal = $total(sales-amount)
>xeq
```

You can total up to ten fields per pass and the $total function is also available in the if command, however, is of limited use.

$Counter Function

For years Suprtool has had the ability to output a record number to an output file with the num option of the output command:

```
>in mysdfile
>out myfile,num,data
```

The above task would generate an output file called myfile, however, you would loose the SD information and you can only put the number at the beginning or the end of the data. Suprtool now has a counter function that allows you to place a $counter at any spot as well as preserve the SD information.

```
>in mysdfile
>def mycount,1,4,double
>ext field1
>ext field2
>ext mycount=$counter
>out myfile,link
>xeq
```

The file myfile will be self-describing and contain the fields field1, field2 and mycount. The field mycount is defined as a double integer, since this is the only field type that the $counter function can use. Each record will have a unique ascending number starting with one.

Bugs Fixed

Bugs Fixed In Suprtool 4.8.02

Get Command. The get command would not get end of file signal properly if the dataset was empty and Set FastRead was on.

Table Command. The information about the data loaded in a Table would be lost if the table data being referenced was the second held table and the previous task involved a chain command.

Bugs Fixed In Suprtool 4.8.01

Chain Command. The chain command would not return the correct record when used with Set FastRead On.

Bugs Fixed In Suprtool 4.8

$Edit Function. The $edit function now returns a proper result when nested within another string function such as $ltrim.

Clean Command. The clean command was improperly upshifting lower case alpha characters.

Bugs Fixed In Suprtool 4.7.12

SubTotal Function. The $subtotal function would fail if the size of the Output buffer was larger than the size of the Input buffer.

Bugs Fixed In Suprtool 4.7.11

Variable Substitution. Suprtool would report the error:

```
Error:  >KEY has 2-4 parms: pos,len[,type][,DESC].
```

when resolving a variable that resolved to a blank line. This is now fixed in Suprtool 4.7.11.

Total Function. The $total function would appear to accumulate incorrectly when sorting in the same task.

SubTotal Function. The $subtotal function would not work if run in the same copy of Suprtool if the previous task used the Duplicate command.

Bugs Fixed In Suprtool 4.7.10

Incorrect Flimit. Suprtool would incorrectly calculate the flimit on an output file when using Numrecs 100%, and if the input file was very large.

$Split generating random characters. The new $split function would put random characters at the point where the split would occur in some cases.

$Split reported bogus error on repeated task. The new $split function would incorrectly report an error in a second task with multiple $split operations.

$Number decimal only numbers incorrect. The new $number function did not handle numbers that consisted of only a decimal place followed by any number of zeroes and a number, as in .01 thru .09.

Empty Join File caused abort. Suprlink would abort if the Join file was empty.

Join File held open. Suprlink would hold the Output file open after the task was completed.

$Subtotal incorrect. Suprtool would give incorrect numbers for a $subtotal function in certain cases.

Bugs Fixed In Suprtool 4.7

Incorrect record Number. Suprtool would incorrectly report the record number when an Illegal Ascii digit was encountered on a Duplicate operation. We no longer attempt to show the record number from the Input source.

Set Limits Tablesize. Suprtool would report an error on a second Set Limits TableSize command if the value given was larger than the previous.

Extract from a Table. Suprtool would incorrectly report an error in some cases when doing an update from a table. The error "Field offset is beyond the input record length" was incorrectly hit if the input file was smaller than the actual length of the table file.

$Counter Function. Suprtool did not correctly reset the $counter variable in between tasks.

Define Commands. The number of Define commands allowed has been increased to 768 defines. The number of defines allowed used to be a variable number dependant on various system settings, this was changed in Suprtool 4.4.10 to be a fixed structure.