Welcome to Suprtool

Welcome to version 4.4.13 of Suprtool -- the HP-UX handyman for Oracle and Allbase databases and fixed-length data files.   Use Suprtool to quickly select and sort data records. Combine multiple data files using Suprlink/UX.   Use STExport to convert fields in a self-describing input file into an output file that can be imported into other applications.

The Suprtool commands are:

Add	Exit	Key	REDO	USERpause
BAse	EXPort	LINk	Reset	Verify
Before	EXTract	List	SELect	Xeq
Chain	Form	LISTREDO	Set	:*MPE or HP-UX command*
Define	Get	Numrecs	SOrt	*=expression*
DELete	Help	OPen	TAble	
DO	IF	Output	Total	
DUplicate	Input	Put	UPdate	
EDit	ITem	Q	Use	

Keywords
Components
Manual
Software Updates
News
Installation
Start
Running
Issues
Commands
Errors And Warnings
Terms
Quick
Support

Keywords

Everything about Suprtool is organized under these keywords:

KEYWORDS	this list of keywords
UPDATES	how to receive regular updates of Suprtool
COMPONENTS	other Suprtool components
MANUAL	printing the manual and using it
NEWS	the latest changes to Suprtool
INSTALLATION	how to install Suprtool
START	quick start with Suprtool
RUN	how to run Suprtool; different run options
ISSUES	technical issues with Suprtool
COMMANDS	explanation of all Suprtool commands
ERRORS	list of error and warning messages
LINK FILES	format of self-describing files
TERMS	explanation of key terms (e.g., database)
QUICK	brief syntax reminders
SUPPORT	how to contact us for product support

Suprtool Components

The Suprtool package consists not only of Suprtool, but also of other programs that perform useful database functions.   These other programs are STExport and Suprlink.

STExport
Suprlink

STExport - Data Export Utility

STExport converts fields in a self-describing input file into an output file that can be imported into different applications.

Use STExport to produce a formatted output file that can be used to import data into databases and applications.

Other databases have different requirements for the format of input data.   You will have to experiment with the various STExport formatting options to find a format that your particular database tool accepts.

Summary of the STExport commands:

Before	Form	Quote	Verify
Columns	Heading	Redo	Xeq
Date	Help	Reset	Zero
Delimiter	HTML	Set	*=expression*
DO	Input	Sign	:*OS command*
Exit	LISTREDO	Spaces	
Floating	Output	Use	

Description

Suprlink - Multidataset Access

Suprlink is a program that works with Suprtool to add "multidataset" capability to Suprtool. Suprlink is not a set of callable routines.   To use it, you must :run Suprlink.Pub.Robelle or use Suprtool's Link command.

Rather than take the regular path to multiple datasets -- random retrieval via IMAGE keys -- with its well-known performance problems, we have chosen to follow a different path: fast serial extracts plus a very efficient merge. The tests that we have performed indicate that this method is often significantly faster than the "official" IMAGE method of chasing down chains and hash synonyms.

To understand what Suprlink does, think of the process of writing a report. Your report program (written in COBOL, RPG, PowerHouse, or some other language) hunts all over the database with DBFIND and DBGET to collect your data.

It would be faster if the report program could just read a sorted disc file with a big record containing all the data necessary for the report, and this is Suprlink's function.   Suprtool can extract the desired fields from the desired records of the sales detail dataset and put them in a disc file.   Then Suprtool can extract the desired fields from the customer master dataset and write them to a second disc file.   If Suprtool sorts both files by customer, Suprlink can "link" them together, producing a third file whose composite record consists of the related fields from both files.   This file is just what we need to feed into the report program.   For example, a sales report program might read a disc file whose records consist of sales transactions plus customer information.   This file has been sorted by customer number and date. If there are several sales for the same customer, the customer information is just repeated in each record.   The report program reads the records, checks for level breaks, and formats and prints the records.

Documentation

Although we do not provide a revised user manual for pre-release versions of Suprtool, we do provide an updated Help file. The Suprtool/UX Help file, however, has not been completely updated to account for all the differences between Suprtool/MPE and Suprtool/UX. The "running" section of the Help file has been updated for Suprtool/UX. You can access it with this Help command:

```
     >help running
```

The Input command documentation has been updated to show the new Suprtool/UX syntax. You can access the Input command documentation with this Help command:

```
     >help input
```

If you just need a quick review of the Input command syntax, you can use Quick Help.

```
     >hq input
```

The user manual covers the Suprtool commands and answers questions that might arise during Suprtool execution. The manual includes a glossary and an index -- refer to them when you have a problem. Everything in the user manual is also available to the on-line Suprtool user through the Help command. For instructions, try:

```
     >help help
```

WinHelp
Change Notice
Notation

WinHelp Documentation

All Suprtool documentation is available in WinHelp format.   See the chapter on installation for instructions to install the WinHelp documentation.   You can also contact Robelle or one of its distributors to obtain an installation disc with the Suprtool documentation in WinHelp format.

Change Notice

For a complete description of the latest changes made to Suprtool, the installation instructions, and any compatibility issues, see the change notice that was included with the release.

Notation

The Suprtool documentation uses a common notation in describing all commands. Here is a sample command definition:

> EXTRACT *field* [(*subscript*)] [=*value*] [,...]

1. UPPERcase letters - literal symbols to be used in the command as they appear (e.g., EXTRACT).

2. Lowercase, underlined or italic - "variables" to be filled in by the user (example: *field*). Each such "variable" is defined elsewhere in terms of literal symbols (consult the index). In the help file, underlining and italics are not available and variables appear simply in lowercase.

3. Brackets - enclose optional fields (example: [(*subscript*)]).

4. Braces - enclose comments in examples. For example, >INPUT ACTREC {input from a data file}. Braces **can** be used for comments in actual Suprtool commands.

5. Up lines - separate alternatives from which you select (example: Set Ignore [On|Off]); sometimes, the alternatives are shown listed on several lines.

6. Dot-dot-dot (...) - indicates that the variable may be repeated many times in the command.

7. Other special characters - literal symbols that must appear in the command as they are shown in the format (example: the , above). Some commas in Suprtool are optional.

In examples, there is an implied carriage return at the end of each line.

Software Updates

As HP continues to release new versions of MPE and HP-UX, you might be wondering how all these operating system changes affect tools such as Suprtool. This is just one reason to have our software support. This service entitles you to many benefits, including regular updates of Suprtool. Each release includes new features, as well as providing support for a changing HP environment. If you are a second site within a large organization, you can either pay for your own support or get support and updates from your internal central site.

News Memos

News Memos

Do you receive a copy of *What's Up, DOCumentation?*, our regular news memo about Robelle, MPE, and HP-UX? We distribute our news memos only to sites with current service. Your copy may be going to your corporate headquarters.

Highlights in 4.4.13

¤ Support for "well-formed" XML in STExport.

¤ Commands entered in a if $read task were logged to the redo file in error.

¤ In some cases a table was not held.

¤ Suprtool, STExport and Suprlink can now have warnings turned off when run from batch with the set warnings off command.

Highlights in 4.4.12

¤ There are no new features in this version of Suprtool.

Highlights in 4.4.11

¤ Some division operations would not work properly if the decimal portion was large and the target type was packed or zoned.

¤ Dividing zero by zero would end up with a large number for some target data types.

¤ Suprtool now handles extracts of a constant on a char type field properly.

¤ A warning message has been added in the case where fields defined as non-standard integers, will be treated as strings.

¤ The error message that prints when the expression specified cannot be coerced/converted into the target has been improved.

¤ Verify Define will now show the correct information when defining non-standard integers.

Highlights in 4.4.10

¤ There are no new features in this version of Suprtool.

Installing Suprtool/UX

Here is an outline of the topics covered:

README	Introduction, warnings, overview
1=ROOT	Log in as root
2=DIRECTORY	Create Robelle directory
3=RESTORE FILES	Restore files from tape
4=ROBELLE VARIABLE	Set the ROBELLE variable
5=WINHELP	Install Winhehlp files (optional)

General Installation Notes

Here we describe how to install and configure Suprtool. The following general notes describe the installation of Suprtool.

Who Should Use These Instructions?

The system manager should use the following installation instructions to install Suprtool/UX. No one can be using Suprtool/UX during the installation. The installation should only take a few minutes.

Summary of Installation Steps

Installing Suprtool involves the following steps:

1. Log in as root.
2. Create the correct directory structure.
3. Restore Suprtool/UX and its associated files from the distribution tape.
4. Set the ROBELLE variable in order to let Suprtool know where its supporting files are.
5. Install WinHelp documentation files. (optional)

Step 1: Log In as Root

There are two ways to log in as root:

a. Exit from HP-UX, and log in with root as the user name.

b. If you are already logged in, you can execute this command:

```
su -
```

In either case, you have to supply the user password for root.

Step 2: Create Robelle Directory

Before restoring files, you must first create the directory where Suprtool/UX will reside:

```
mkdir /opt/robelle
```

Step 3:   Restore Files

Use the following command to restore the Suprtool/UX files from the distribution tape:

```
     tar xv /opt/robelle
```

This command assumes your tape device is /dev/rmt/0m.   If it is not, you need to specify your tape device using the f option in the tar command.   For example, if your tape device is /dev/rmt/1m, you need to use the following command to restore the files:

```
     tar xvf /dev/rmt/1m /opt/robelle
```

HP-UX 10.0 and 10.10

Your tape has two copies of Suprtool.   The version in /opt/robelle runs only on HP-UX 10.20 and later.   The version of Suprtool in the /usr/robelle directory is compatible with HP-UX 9.*x* to 10.10.

The naming convention for third-party programs on HP-UX has changed from /usr to /opt.   If you want to use Suprtool in the /opt directory, you need only the following commands:

```
     mv /usr/robelle /opt
     ln -s /opt/robelle /usr/robelle
```

Step 4: Set the ROBELLE Variable

You must set the ROBELLE environment variable so that Suprtool can find its help and suprmgr files. How you set this variable depends on where you have installed Suprtool.

Bourne and Korn Shells

```
    export ROBELLE=/opt/robelle
```

C Shell

```
    setenv ROBELLE /opt/robelle
```

If you install Suprtool in any other directory, you must set the ROBELLE environment variable to that directory.

Details of how to set up the PATH and MANPATH variables are included in the chapter "Running Suprtool Under HP-UX."

Installing WinHelp Documentation Files (optional)

The Suprtool documentation is available in the WinHelp file format of Microsoft Windows.

The WinHelp files may be found in a number of places:

1. On a CD included with the printed Suprtool manual.
2. On your system, installed when you installed Suprtool.
3. On the Internet at `http://www.robelle.com/library/manuals/`.

Installing From the CD

1. Insert the Suprtool Help CD into your CD drive.
2. Click the **Start** button and then click **Run**.
3. Type **<drive>:setup**. For example, if the disk is in drive M, type **m:setup**.
4. Follow the instructions on your screen.

Installing from Your System or the Internet

If you do not have the CD, you can get the files from the Internet or from your system. In either case, once you have the files on your PC, the installation instructions are the same.

Make a Temporary Directory for the Setup Files

Create a directory for the installation file on your PC. You can remove this directory after installing the WinHelp files. At the DOS prompt, for example, type

```
mkdir \robtemp
```

Download the Compressed File

Use a web browser to download the Suprtool/UX manual from

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

Download suprux.exe to the directory you created in the previous step.

Or use Reflection to download the self-extracting file to your PC. This file on the HP 9000 is called

```
/opt/robelle/winhelp/suprhelp.exe
```

Log on to the HP 9000 and press Alt-Y to go to the Reflection Command Line window. Then download the file as a binary transfer.

```
receive c:\robtemp\suprhelp.exe
       from /opt/robelle/winhelp/suprhelp.exe binary
```

Expand the Compressed Files

Locate the file you downloaded and run it.   E.g., from the DOS prompt, type the following commands:

```
cd \robtemp          {go to the new Robtemp directory}
suprhelp.exe          {extract the Help files}
```

This expands the downloaded program into a Setup program and all its associated files.

Run the Setup Program

Run the Setup program.   E.g., from the DOS prompt enter

```
setup.exe             {install the Help files}
```

Follow the instructions in the Setup program.

You can now access all of the Suprtool documentation by selecting any of the icons in the "Robelle Help" Start Menu folder.

Remove the Robtemp Directory

Before you remove the temporary files, you can copy them to a diskette and distribute them to other Suprtool users.

To delete all the temporary files and remove the Robtemp directory, use either the Windows Explorer or the following DOS commands:

```
cd \robtemp
del *.*              {delete all the files in Robtemp}
cd ..
rmdir robtemp        {remove the Robtemp directory}
```

Getting a Quick Start with Suprtool

This section of the on-line help gives you examples of some common Suprtool tasks.

How to Run Suprtool

Use the following command to access Suprtool:

```
/opt/robelle/bin/suprtool

SUPRTOOL/Copyright Robelle Solutions Technology Inc.  1981-2001.  (Version
4.4.13)  SUN, JUN 17, 2001, 11:47 AM  Type H for help.  Licensee:  The Shum
Co.  Today's Hint.  To see ALL of the options available in Suprtool, use
>VERIFY ALL.  >
```

Suprtool prints its version number and the current time right after a banner. It also prints the name of the company that has licensed this copy of Suprtool.

Suprtool then prompts with ">".   Press Return after typing each command.   For example, if you type the help command:

```
     >help
```

Suprtool prints some help text and a keyword list.   Type a keyword or press Return to leave Help.

To exit Suprtool, type "Exit" at the Suprtool prompt.

```
     >exit
```

What is a Task?

Tasks are the building blocks with which Suprtool helps you to solve data processing problems.   In a task, Suprtool reads information from a file or database, selects and processes some information, and writes out the result. You can visualize a Suprtool task like this:

```
     +----------------+
     |                |       Suprtool          +----------+
     | Input Records  | ----> selects and ----> | Output   |
     |                |       processes         | Records  |
     |                |                         +----------+
     +----------------+
```

The examples that follow all consist of Suprtool tasks.   Simple solutions require only one task.   Complex solutions consist of several tasks, often with the output of one task becoming the input for the next task.

Copying Files

Copying One File

Use the Input command to specify a data file.

```
>input  file1,reclen 80, nolf
>output result
>xeq
```

The Output command creates the file called "result", which is a copy of the input file.

Appending to a File

To append to an existing file, use the Append option in the Output command.

```
>input  file2, reclen 80, nolf
>output result,append
>xeq
```

Fields in Data Files

SD file
Create an SD file
Creating Fields
Data File to SD file

What is a Self-Describing File?

A self-describing (SD) file is actually a pair of files, one with data and the other with field information.   These files have the advantage of behaving like data files, which can provide field information to Suprtool without having to Define all the fields.   The Input command is also simpler because there is no need for either the Reclen or the LF parameters.

Creating an SD File

To create an SD file, use the Link option in the Output command.

```
>select * from sales
>output result,link
>xeq
```

Now the data file "result" has the same field names as the Sales table. Suprtool can read this data file and know about the fields automatically.

```
>input  result
>if     sales_total>20000 and product_price<5000
>output custlist
>xeq
```

Define Fields in a Data File

A regular data file does not have any field information associated with it. If you need to work with the fields in a data file, you need to tell Suprtool about the fields using the Define command.   For example, say you have a data file with lines that look like this:

```
12345678John        Rutherford     <32>
98765432Catherine   Smith          <29>
|       |           |               |
Account First name  Last name       Age
8-byte  12-byte     16-byte         2-byte integer
```

Use these Define commands to tell Suprtool about the fields:

```
>input  datafile, reclen 38, lf
>define account,    1,  8, byte
>define first_name, 9, 12, byte
>define last_name, 21, 16, byte
>define age,       37,  2, int
         |         |    |    |
     field name    |  Length |
                 Start       Data-type
                 position
```

Now you can use the field names "account", "first_name", "last_name", and "age" to refer to the corresponding parts of the line, just as if this was a database record.

```
>input  datafile, reclen 38, lf
>define ...
>if     age>65
>ext    account, last_name
>output result
>xeq
```

Create an SD File from a Data File

To create an SD file from a data file, follow these steps:

1. Define the fields that you want to include in the SD file.
2. Extract the fields in the order you want.
3. Use the Link option in the Output command to create the SD file.

```
>input   datafile, reclen 38, lf
>define  account,    1,  1, byte
>define  first_name, 9, 12, byte
>define  last_name, 21, 16, byte
>define  age,       37,  2, int

>extract account, age, first_name, last_name

>output  result,link
>xeq
```

Repeating Commands

Repeat a Command

Repeating a Command

Use the Listredo command to see a list of your most recent commands. Use the Do command to repeat a command, or use the Redo command to modify a command before repeating it.

```
>listredo               {20 most recent commands}
>listredo input         {most recent Input commands}

>input result
>...
>xeq
>do  input              {repeat previous Input command}

>input result
>if  quantity > 10000
>...
>redo if                {modify previous If command}
                        {then repeat}
```

If you have used two commands that begin with the same letter, you have to be careful when repeating those commands. Make sure you use enough letters to identify each command distinctly from the other. In the following example, if you wish to repeat the Input command instead of the If command, you need to use "do i s" instead of just "do i".

```
>i somefile
>if <expression>
>...
>xeq
>do i s                 {repeats previous Input command}
```

Selecting Database Records

These examples show you how to get records from an Oracle table.   They assume you have opened the database with the Open command.   The results are written to a data file called "result", which can be read either by a program or by a report writer.

```
     $/opt/robelle/bin/suprtool
     >open oracle demo reader
```

All Records
First Few Records

Select All Records

This example extracts all the records from the dataset.   Note that we didn't specify any selection criteria, so Suprtool selects all the records.

```
    >select * from sales  {input table}
    >output result        {output file}
    >xeq                  {Xeq command performs the task}
```

Look at the First Few Records

If you want to look at the first few records of a dataset, use the Numrecs command. This command tells Suprtool to extract at most the number of records specified. Then, instead of sending the result to a file, send it to the screen with "output *,ascii". The example shows you how to look at the first 10 records in your dataset.

```
>select * from sales
>numrecs 10                {first 10 records}
>output  *,ascii           {output to screen, format numbers}
>xeq
```

Selecting by Criteria

Simple Criteria

To tell Suprtool to choose records based on certain criteria, you can either use any valid SQL command (e.g., select, where), or you can select all the records and use an If command. In these two examples Suprtool extracts all records with a sales_total value greater than 20000 from the Sales table. Both tasks produce identical results, but one way may be faster than the other.

```
>select * from sales where sales_total > 20000
>output result
>xeq

>select * from sales
>if     sales_total > 20000
>output result
>xeq
```

Complex Criteria

To choose records using more complex criteria, combine several simple criteria using AND, OR, NOT, and parentheses.   In this example Suprtool extracts all records that have a sales_total value greater than 20000 and a product_price value less than 5000 from the Sales table.

```
>select * from sales
>if     sales_total>20000 and product_price<5000
>output result
>xeq
```

String of Digits

If you have a byte-type field that consists entirely of digits, and you want to use this field as a numeric field, you need to define a new display field on top of the existing field. For example, suppose your data looks like the following, where the customer account number is stored in the 8-digit byte-type field at the start of the record:

```
20476789...rest of customer record...
```

To find all customers with an account greater than 20470000, you would do the following:

```
>select * from table
>define accountnum, account, 8, display
>if     accountnum > 20470000
>output result
>xeq
```

Selecting by Date

The following section on dates does not apply to SQL columns, only to defined fields and SD fields. Disc files usually store dates as numeric or character fields; you can use the Define command to isolate the field.

Before Suprtool can use a date field, it has to know the format of a particular date field. Use the Item command to specify the format. For example, to tell Suprtool that the item purch_date is a date field with a format of yyyymmdd (e.g., 20000319), you would use:

```
     >define purch_date, 11,8             {8 bytes, starts in byte 11}
     >item   purch_date, date, yyyymmdd  {date format}
```

In the following date examples, we show the Define and Item commands in each example. In practice, however, you only need to use these commands once per date field, not once per task.

Select by Today's Date

For this example, select all the sales records whose purchase date is today. Note the use of $today in the If command to indicate today's date.

```
>input  saledata,reclen 70,nolf
>define purch_date, 11,8
>item   purch_date,date,yyyymmdd
>if     purch_date = $today       {select today's date}
>output result
>xeq
```

Other tricks with $today:

```
>if     purch_date = $today(-1)   {yesterday}
>if     purch_date = $today(+1)   {tomorrow}
```

Select by Particular Date

To specify a particular date, use the $date function in the If command. The $date function has the form $date(*year /month /day*). This example selects all the sales transactions for August 12, 2000.

```
>input  saledata,reclen 70,nolf
>define purch_date, 11,8
>item   purch_date,date,yyyymmdd
>if     purch_date = $date(2000/08/12)
>output result
>xeq
```

Select by Year

Suppose we want to select all the sales transactions for 2000. Suprtool does not have a short-hand for specifying "everything in that year". To specify an entire year, use a date range from January 1st to December 31st.

```
>input  saledata,reclen 70,nolf
>define purch_date, 11,8
>item   purch_date,date,yyyymmdd
>if     purch_date >= $date(2000/01/01) and &
        purch_date <= $date(2000/12/31)
>output result
>xeq
```

Select Prior Month

In the $date function, use a * to indicate the current year, month, or day. Similarly, a *-1 means the previous year, month, or day.   For this example, select all the sales transactions for the prior month.   Note the use of the special keywords "first" and "last" to indicate the first and last day of the month.

```
>input  saledata,reclen 70,nolf
>define purch_date, 11,8
>select * from sales
>item   purch_date,date,yyyymmdd
>if     purch_date >= $date(*/*-1/first) and &
        purch_date <= $date(*/*-1/last)
>output result
>xeq
```

Selecting by Lists of Values

Sometimes you want to find records based on criteria contained in another file or table.

Finding Data Based on a List

Suppose we want to find all orders for the customers "1234", "9876", and "5555."   We simply use a list of values after the equal sign in the If command.   A match is made if a customer matches any one of the values in the list.

```
>select * from order_details
>if     cust_no = "1234", "9876", "5555"
>output orders
>xeq
```

If we wanted to find orders for all customers except "1234", "9876", and "5555", we would simply change the equal sign in the If command to a not-equal sign.   A match is made if a customer does not match any values in the list.

```
>select * from order_details
>if     cust_no <> "1234", "9876", "5555"
>output orders
>xeq
```

Finding Data Based on a File

If you have a large list of values in a file, you can load them into Suprtool and select data based on this list.   First use the Table command to load values from an external file into a table.   Then use the $lookup function of the If command to match data to the table.

Suppose our list is in a self-describing file called Custlist.   We create a program-temporary table called cust_table.   Note that this is not the same as an Oracle table.

```
    >select * from order_details
    >table  cust_table, cust_no, file, custlist
    >if     $lookup(cust_table, cust_no)
    >output orders
    >xeq
```

If you want to find all customers not on the list, just negate the If condition.

```
     >if    not $lookup(cust_table, cust_no)
```

Finding Data Based on Another Table's Criteria

Sometimes you need to find data from one table based on conditions from another table. This is a typical example: you want to find all of the pending orders for those customers whose accounts receivable balance is 0.

First we find the customers with an AR balance of 0 and extract their customer numbers to a self-describing file.

```
>select * from receivables
>if     ar_balance = 0
>ext    cust_no
>output custlist,link
>xeq
```

Now we can find information by loading a file of customer numbers into a table and then applying the $lookup feature.

```
>select * from order_details
>table  cust_table, cust_no, file, custlist
>if     status="PENDING" and $lookup(cust_table,cust_no)
>output orders
>xeq
```

Finding Data in a Data File

So far, the examples have looked up data from a table.   If you want to look up information in a data file, you need to tell Suprtool about the fields.   Use the Define command to do this.

The following example gives you some idea of the byte-size of one kind of record in a data file.

```
John        Smith           12345678
Anna-May    Richardson      98765432

12-bytes    16-bytes        8-bytes
```

If you want to look up customers based on a list of customer numbers in the self-describing file Custlist, use the following task.   Notice how the start position and number of bytes is entered into the Define command.   This defines the position within the input file, not the table file.

```
>input  flatfile, reclen 36, nolf
>define cust_no, 29, 8, byte
>table  cust_table, cust_no, file, custlist
>if     $lookup(cust_table, cust_no)
>output result
>xeq
```

Sorting Database Records

Sort Records

To tell Suprtool to sort table records, you can either use any valid SQL command (e.g., select, order by), or you can select the records and use a Sort command.   Here are two examples where Suprtool extracts all records from the Sales table into a data file called "result".   The records are sorted by the field cust_account.   Both tasks produce identical results, but one way may be faster than the other.

```
>select * from sales  order by cust_account
>output result
>xeq

>select * from sales
>sort   cust_account
>output result
>xeq
```

Sort Records in Descending Order

This example extracts all records from the Sales table into a data file called "result".   The records are sorted by the field sales_total in descending order (i.e., show highest totals first). Use the Desc option in the Sort command to do this.

```
>select * from sales
>sort   sales_total desc  {descending order}
>output result
>xeq
```

Sort by Multiple Keys

This example extracts all records from the Sales table into a data file called "result". The records are sorted by the field cust_account, then by sales_total in descending order. Use two Sort commands to do this because the Sort command only accepts one field at a time.

```
>select * from sales      {input}
>sort   cust_account      {first sort key}
>sort   sales_total desc  {second sort key}
>output result            {output}
>xeq
```

Duplicate Records

In the following examples, the key field is in the first four bytes of the record.   "Duplicate-ness" is based on records having the same key value.   In any group of records with the same key value, the first record is considered to be the "original", and the rest are considered to be the "duplicates".

Report without Duplicate Records

This is an example of filtering out duplicated records (the original remains). This is done by using the None option of the Duplicate command.

```
Input      Result

1111 a     1111 a
2222 b     2222 b
2222 c     3333 e
2222 d
3333 e

>select * from table
>sort   keyfield
>dup    none keys
>output result
>xeq
```

Report Only the Duplicate Records

This is an example of keeping only the duplicated records (the original is not kept). This example is the opposite of the previous example. Use the Only option of the Duplicate command to do this.

```
Input      Result

1111 a     2222 c
2222 b     2222 d
2222 c
2222 d
3333 e

>select * from table
>sort   keyfield
>dup    only keys
>output result
>xeq
```

Report Only the Unique Records

This example shows how to report only those records without duplicates. That is, if the records have duplicates, both the originals and their duplicates are omitted from the report.

```
     Input        Result

     1111  a      1111  a
     2222  b      3333  e
     2222  c
     2222  d
     3333  e
```

You have to use two Suprtool tasks to accomplish this. The first task creates an intermediate file Dupfile that contains the keys of the duplicate records. The second task creates the desired output file Result that contains only the unique records.

```
     >select  * from table
     >sort    keyfield
     >extract keyfield
     >dup     only keys
     >output  dupfile, link
     >xeq

     >select  * from table
     >table   dup_table, keyfield, sorted, dupfile
     >if      not $lookup(dup_table, keyfield)
     >output  result
     >xeq
```

Report Only the Duplicates and Their Originals

This performs the opposite function to the one outlined above. It keeps only the duplicates and their originals.

```
Input        Result

1111  a      2222  b
2222  b      2222  c
2222  c      2222  d
2222  d
3333  e
```

Once again, you have to use two Suprtool tasks to accomplish this. The first task creates an intermediate file Dupfile that contains the keys of the duplicate records. The second task creates the output file Result that contains only duplicate files and their originals.

```
>select  * from table
>sort    keyfield
>extract keyfield
>dup     only keys
>output  dupfile, link
>xeq

>select  * from table
>table   dup_table, keyfield, sorted, dupfile
>if      $lookup(dup_table, keyfield)
>output  result
>xeq
```

Deleting Duplicate Data File Records

The following tasks read the file Datafile and create two new files. The file named "result" does not have duplicate records. The other file named "archive" has only the duplicate records.

Task 1: Identify which records to delete.

```
>input  datafile,reclen 38, nolf
>define key1,1,8
>define key2,13,12

>define rec,1,38     {length of the record}
>ext    rec
>sort   key1
>sort   key2
>dup    only keys
>out    dupfile, link
>xeq
```

Task 2: Write records to archive.

```
>input  datafile, reclen 38, nolf
>table  duptab, rec, sorted, dupfile, hold
>if     $lookup(duptab, rec)
>output archive
>xeq
```

Task 3: Delete the records.

```
>input  datafile, reclen 38, nolf
>if     not $lookup(duptab, rec)
>output result
>xeq
```

Decimal Places

The following section on dates does not apply to SQL databases, only to disc files.

Data in disc files often has an implied number of decimal places. For example, dollar amounts usually have two implied decimal places for the cents. In this case, the number stored is scaled by a factor of one hundred (e.g., you would enter 10000 to represent $100.00).

```
>input  saledata, reclen 70, nolf
>def  total_sales, 40, integer
>if   total_sales > 99900  {find sales > $999}
>out  result
>xeq
```

You can use Suprtool's Item command to identify defined fields that have an implied number of decimal places.

You can use Suprtool's Item command to identify defined fields or database items that have an implied number of decimal places. Once you do this, you can then enter regular, unscaled numbers. For example, to enter five cents, use 0.05; to enter $100.00, use 100. If a field is a dollar and cents amount scaled by 100, use the following to tell Suprtool about the decimal place:

```
>item total_sale, DECIMAL, 2
```

With the Item command, the previous example becomes more understandable:

```
>input  saledata, reclen 70, nolf
>def  total_sales, 40, integer
>item total_sales, decimal, 2
>if   total_sales > 999  {find sales > $999}
>out  result
>xeq
```

Converting Numbers

There are several ways to convert binary numbers (e.g.   I2, P8) into human-readable ASCII form.   You can use STExport's Output,ASCII or Output,DISPLAY if you want to convert all numbers.

If you want to convert only some of your numeric fields, you can use Suprtool's numeric conversion feature to convert the binary fields to display fields.

```
define mynumber,1,6,display
get dataset
ext some-fields...
ext mynumber = binary-number
output filename
xeq
```

Note that this technique also works for converting a number from one numeric type to another numeric type.

Counts and Subtotals

Count and Subtotal on Sort Keys

This example counts the number of sales transactions for each customer and produces the total sales for each customer.   We use the Count and Total options of the Duplicate command.   Note that we made the output file self-describing so we can easily work with it later.

```
>input transactions            {self-describing file}
>ext  cust_account
>sort cust_account             {need to sort by key}
>dup  none keys count total sales_total
>list standard
>out  result, link
>xeq
```

The output file contains three fields.   The first field is the cust_account that we extracted. Suprtool created two new fields at the end of each output record: st-count and st-total-1. St-count contains the number of times each cust_account occurred, while st-total-1 contains the total sales for each cust_account.

Sort by Count or Subtotal

When Suprtool counts or subtotals, the output is sorted according to the key fields. If you want your output file to be sorted by the count or by a total, you must process the output file with a second task. The following example sorts the previous file of totals by ST-COUNT. We choose a descending sort sequence, so that we can see first the customers with the largest number of orders.

```
>input result              {input from previous task}
>sort  st-count, desc      {highest counts appear first}
>list  standard            {produce a simple report}
>xeq
```

Total by Field

If you want to get a single total for a field, without caring about subtotals on sort breaks, you can use the Total command.   Total prints out the result on $stdlist.   For example, to compute the total sales value for 2000 transactions, use these commands:

```
     >select * from sales
     >if     purch_date>=000101 and purch_date<=001231
     >total  sales_total
     >output $null
     >xeq
```

Listing Records

You can print selected input records either formatted or with the Octal, Hex, Decimal, or Character representations.   To dump all sales records with a negative amount, use these commands:

```
>select * from sales
>if sales_total < 0
>list lp
>xeq
```

This finds the entries that meet the selection criteria and prints them to the default line printer, showing column names and column values converted to ASCII.   If you suspect that your data is bad, you can dump the records in Octal/Char format instead:

```
>select * from sales
>if sales_total < 0
>list octal,char
>xeq
```

If you want the listing in column format, use List Standard:

```
>select * from sales
>if sales_total < 0
>list standard lp
>xeq
```

Changing the Output Record Format

You can change the output file record format by using the Extract command. The Extract command causes Suprtool to assemble Output records by stringing together fields extracted from Input records. You would use the following to extract two of the nine fields from the customer records:

```
>select * from customer    {input from a table}
>extract cust_account      {extract the key value and}
>extract credit_rating     {    one other field}
>output out1               {output file will have two}
>xeq                       {    fields}
```

You can easily insert data into the middle of a record, again using the Extract command. Define the first and second halves of the record as two big chunks. Now Extract the first part, note the constant you wish to insert, then Extract the second part.

```
>input myfile, reclen 95, nolf      {95 bytes wide}
>define part1,1,60,byte             {first 60 bytes}
>define part2,61,35,byte            {remaining 35}
>extract part1, "constant", part2
>output newfile                     {103 bytes now}
>xeq
```

Producing a Condensed Table Listing

When debugging test databases, it is often desirable to produce a condensed listing of a table on $stdlist.   The following example combines the Extract command with the ASCII output option (i.e., all binary and packed-decimal data is converted into readable ASCII characters).   For readability, each data value is prefixed with an abbreviated column name. This listing is more compact than the one produced by the List command.

```
>select * from customer
>extract "Account=",cust_account," "
>extract "C/R=",credit_rating
>output *,ascii      {* implies $stdlist}
>xeq
```

The output would look like:

```
Account=04598921 C/R=    500000
Account=44657844 C/R=   2000000
Account=98753198 C/R=    300000
```

Simple Reports

You can produce simple reports with Suprtool's List command.   You select the records for the report with the If command and the fields for the report with the Extract command. Reports can include running headings with the date, title, and page number and an optional line of column headings.   Suprtool can produce default titles and headings.

```
>select  * from customer
>extract cust_account
>extract credit_rating
>list    standard
>xeq
```

The output would look like this:

```
Jan 17, 2000 11:59                                   Page 1

CUST_ACCO CREDIT_RATING

 4598921       5000.00
44657844      20000.00
98753198       3000.00
```

Sample Report
Titles
Column Headings
Mailing Labels

Your First Report

Our report selects all customers in California, sorts the records by city, and reports on the city, account number, and name of each California customer:

```
>select  * from customer    {input dataset}
>if      state = "CA"       {California customers}
>sort    city               {sort by city name}
>extract city               {city first on each line}
>extract cust_account       {followed by account#}
>extract name_first         {and first name}
>extract name_last          {and finally last name}
>list    standard           {produce a quick report}
>xeq
```

These commands produce a report with four columns. The title consists of the date and page number. The column headings are the name of each column that we extracted.

Specifying Your Own Title

If you don't like Suprtool's default title, you can override it with your own:

```
     >list standard,title "Customers in California"
```

Column Headings

Column headings default to uppercase field names. The names are truncated if they are longer than the field itself. One space is inserted between fields.

Suprtool does not automatically align user-specified headings with the data columns. We suggest specifying heading strings with the same length as the fields they represent, while taking into account the space between the data columns.

In our example, we enter one column heading per line (using Suprtool's continuation character "&"):

```
>list standard,heading  &
      {1...5....10...5}
      "City          " &    {X12}
      "Account   "     &    {Z8}
      "First Name "    &    {X10}
      "Last Name"           {X16}
```

We included one space between fields. Note that an extra space was needed for the `Account` heading (it is an 8-digit field, but we used 10 characters). Because cust-account is a zoned-decimal field, an extra space is required for the sign.

Printing Mailing Labels

You can print mailing labels by combining the Extract command with the List Oneperline command.   We assume that each mailing label starts with two blank lines, followed by the customer name and address, followed by another blank line.   The Suprtool commands to produce the labels are as follows:

```
>select  * from customer    {input customers}
>extract " "                {first field}
>extract " "                {second field}
>extract customer_name      {name first}
>extract street_address1    {three lines of address}
>extract street_address2
>extract street_address3
>extract " "                {last blank field}
>list    oneperline, noname, noskip, norec
>xeq
```

The `extract " "` line creates a single field that consists of a blank space. Each of these blank fields results in a blank line on our mailing labels, since the List command puts one field on each output line.

If you want to combine two fields on one line, you would first have to create an output file with the combined fields and use this file as input to List Oneperline.

Running Suprtool/UX

"How to access Suprtool/UX" is organized under these keywords:

RUN	how to run Suprtool/UX
SHELLS	configuring different shells
PATH	suggested path for using Suprtool/UX
CONTROL	configuring tty control characters
SUPRMGR	customizing Suprtool for yourself
BATCH	Suprtool's assumptions for batch mode running
OPTIONS	command line options
HP-UX	HP-UX specific features
MPE VS. HP-UX	differences between Suprtool/MPE and Suprtool/UX

Running Suprtool

To run Suprtool for HP-UX, type this command:

```
  /opt/robelle/bin/suprtool

  Suprtool. Copyright Robelle Solutions Technology Inc. 1981-2001.
  (Version 4.4.13) Type ? for help.
  >
```

Suprtool prints its version number and prompts with ">". Press Return after each command you type. For example, if you type

```
     >help
```

Suprtool prints some Help text and a keyword list. Type a keyword for more specific information, or press Return to leave Help.

Configuring Different Shells

When you log in to HP-UX, a shell program is invoked.   This program interprets commands, executes them, and controls command execution.   To make configuration changes, you have to know which shell you are using and what files are automatically executed.

Bourne and Korn Shells

The Bourne and Korn shells execute the file /etc/profile when you log in to HP-UX.   Then they look for a .profile file in your home directory.   If it exists, the file is executed.   If you use SAM to add new users, the file /etc/d.profile is automatically copied to the home group of the new user.   If you want to make global changes to the commands executed at login time, you should change two files:

`/etc/profile`	{always executed at login time}
`/etc/d.profile`	{default .profile for new users}

C Shell

The C shell executes the file /etc/csh.login when you log in to HP-UX. First it looks for the .login file in your home directory. If the file exists, it is executed. Then the C shell executes the *.cshrc* file in your home directory (also executed any time you invoke a new copy of /bin/csh). If you use SAM to add new users, the files /etc/d.login and /etc/d.cshrc are automatically copied to the home group of the new user. If you want to make global changes to the commands executed at login time, you should change these files:

`/etc/csh.login`	{always executed at login time}
`/etc/d.login`	{default .login for new users}
`/etc/d.cshrc`	{default .cshrc for new users}

Setting Up a PATH for Suprtool

You can invoke Suprtool with the command:

```
/opt/robelle/bin/suprtool
```

If you just type

```
suprtool
```

to invoke Suprtool/UX, you must either add /opt/robelle/bin to your PATH or copy /opt/robelle/bin/suprtool to a directory that is currently on your PATH. Similarly, the man pages for Suprtool are in /opt/robelle/man/man1/suprtool.1. To make the man pages available to everyone, you can either add /opt/robelle/man to your MANPATH or you can copy the man pages to a directory that is currently on your MANPATH.

Bourne And Korn Shells
C Shell

Bourne and Korn Shells

See Configuring Different Shells (above) for a discussion on the files that are automatically executed by the Bourne and Korn shells.   The easiest way to change the two PATHs for all the users on your HP-UX machine is to logon as root, and add these two lines to the file /etc/profile after any existing PATH or MANPATH statements:

```
PATH=$PATH:/opt/robelle/bin
MANPATH=$MANPATH:/opt/robelle/man
```

Remember to delete any PATH or MANPATH settings in /etc/d.profile so that new users do not override your changes.   You also have to warn existing Bourne and Korn shell users to change the .profile file in their home directories.

C Shell

See Configuring Different Shells (above) for a discussion on the files that are automatically executed by the C shell.   The easiest way to change the two PATHs for all the users on your HP-UX machine is to logon as root, and add these two lines to the file /etc/csh.login after any existing PATH or MANPATH statements:

```
set path ($path /opt/robelle/bin)
setenv MANPATH "$MANPATH":/opt/robelle/man
```

Remember to delete any PATH or MANPATH settings in both /etc/d.login and /etc/d.cshrc so that new users do not override your changes.   You also have to warn existing C shell users to change their .login and .cshrc files in their home directories.

Control Characters and stty

Most HP-UX users have Control-D configured as the end-of-file character, and Control-C as the interrupt character. If you use Robelle-style modify, you must reassign Control-D to a different control character. If you are familiar with MPE, you may want to assign Control-Y as your interrupt character. A standard shell configuration file (.profile for Bourne and Korn shells, and .login for the C shell) usually contains a line like:

```
    stty erase "^H" kill "^U" intr "^C" eof "^D" swtch "^Z"
```

To change both the end-of-file and interrupt characters, you should change the "intr" and "eof" control keys as follows:

```
    stty erase "^H" kill "^U" intr "^Y" eof "^E" swtch "^Z"
```

Note that many programs require an end-of-file signal. Many introductory books on UNIX assume that Control-D signifies the end-of-file. Once you have changed the control character, remember to use Control-E for the end-of-file (at least Control-E is easy to remember since end-of-file starts with the letter "E").

Suprmgr Configuration Files

When you run Suprtool, it automatically "uses" this configuration files if it exists:

```
     /opt/robelle/suprmgr
```

The system manager usually creates /opt/robelle/suprmgr, and puts Suprtool commands in it to set Suprtool options.   To check the options for your site, examine this configuration file.

On-Line vs. Batch Access

You normally run Suprtool as an on-line session. You type Suprtool commands on your terminal, and Suprtool prints responses on your terminal. If you redirect stdin or stdout, Suprtool assumes that it is in batch.

Suprtool in batch is almost identical to Suprtool on-line, except for answering questions. When Suprtool asks a question in batch, it does not expect an answer from stdin because no one is there to answer. Suprtool assumes that you want your batch task to complete, so it always selects the option that completes the command successfully. This is normally a "yes" answer, as in "yes, purge the file". Suprtool prints the question on stdout, as well as the answer that it has selected for you.

Command Line Options

You can invoke Suprtool/UX with various options.   The syntax for invoking Suprtool/UX is

```
     suprtool [-cv]
```

Initial Command Line: -ccmdstring

You can specify commands by using the -c option followed by the actual commands.   There must be no space between the -c and the command list.

If there is a space within the command, the whole command must be enclosed in single or double quotation marks; otherwise, the quotation marks are optional. Here are some examples:

```
suprtool -c"use usefile"
suprtool -c"set prompt $"
```

Default Outcount File Name: -oc

If you want to know how many records Suprtool has processed, use the -oc option.   This option sets the file name for outcount to ".stoutcount".   After a successful task, Suprtool writes the number of output records to the .stoutcount file.   You can then use this file in shell scripts to check for specific record counts.

For example, suppose that you want to check for at least ten records from an Oracle database.   You would write a shell script like:

```
#!/bin/sh
#
#  Select records from an Oracle table and check that there
#  are at least ten.

suprtool -oc << !EOD
open   oracle scott tiger
select * from emp
if     sal > 1000.00
output /dev/null
exit
!EOD
if [ `cat .stoutcount` -ge 10 ]; then
   echo "More than 10 records found"
fi
```

Exit with Verify: -v

Some users inadvertently Exit from Suprtool by entering the Exit command instead of Xeq. To prompt for Exit approval, use the -v option.

```
suprtool -v
>e
Okay to exit [no]:
>
```

Combining -c and -v

You can combine both the -c and -v options with the following command:

```
suprtool -c"use usefile" -v
```

HP-UX Notes

This section describes Suprtool/UX features that interact with the HP-UX environment.

SHELL COMMANDS	executing shell commands
FILENAMES	hardcoded file names

Shell Commands

Shell Commands

You can execute shell commands by typing them anywhere you type a Suprtool command. If a command is both a shell and Suprtool command, you must precede the shell command by an exclamation mark (!) or a colon (:). Shell commands are executed by your default shell (the one configured in /etc/passwd for your user name).

```
$suprtool
>whoami           {these 3 commands are identical}
>!whoami
>:whoami

>set  ...         {does Suprtool's Set command}
>!set ...         {does HP-UX's Set command}
```

Shell commands are executed by a child copy of your shell. Child shells cannot change environment variables in the parent's environment. To change the value of an environment variable, you must first exit Suprtool.

Hardcoded File Names and ROBELLE Variable

Some file names are hardcoded into Suprtool. This section describes the hardcoded file names that Suprtool/UX may need. Suprtool will normally look for files in the /opt/robelle directory unless you set the ROBELLE variable.

ROBELLE Variable

Normally Suprtool looks files in the /opt/robelle directory. If you move Suprtool you must set the ROBELLE variable.

```
     export ROBELLE="/users/robelle"
```

/opt/robelle/suprmgr

This is an optional file that is designed to contain configuration commands. You cannot change this file name. If you move Suprtool/UX to a different directory you must set the ROBELLE variable so Suprtool may find this file.

For example, if you move Suprtool to the /users/robelle directory you must set the ROBELLE variable in the following manner:

```
     export ROBELLE="/users/robelle"
```

You can then put your suprmgr file in the /users/robelle directory.

/opt/robelle/help/suprtool

This is the name of the Suprtool/UX Help file. You can override this name by using Set Filename Help or set the ROBELLE variable as outlined previously.

```
     /set filename help /usr/local/help/suprtool
```

Outcount File

If you want to automatically check the number of output records that Suprtool produced, you must produce an outcount file. This file contains a string with the number of output records that Suprtool processed.

By default, no outcount file is produced. If you invoke Suprtool/UX with the "-oc" option, Suprtool writes the number of output records to a file called ".stoutcount." Use Set Filename Outcount to specify your own file name for the output count.

If you add Set Filename Outcount to the Suprmgr file, every successful invocation of Suprtool/UX produces a file. While these files are small, they may clutter up a busy system so that is why the default file name is none.

Differences Between MPE and HP-UX

We have tried to make the MPE and HP-UX versions of Suprtool as compatible as possible. This section describes how Suprtool/UX is different from Suprtool/MPE.

Record Length

On MPE, Suprtool can obtain the record length of a file. There is no concept of record length on HP-UX because a file consists of a string of bytes. In Suprtool/UX, there are two ways to determine the record length.

1. Specify the record length with the Rec parameter of the Input command.

2. Use self-describing files.

If the specified record size is incorrect, Suprtool/UX cannot verify it. The most common symptom of an incorrect specification in size is an offset of one or more characters in each field.

Line Feeds

In MPE, there is no separator between records in a file. In HP-UX, there may not be a separator, or there may be a line feed between each record. For Suprtool to correctly read a data file, it must know whether the line feeds are present. You can specify whether or not a file has line feeds via the LF or NOLF options in the Input command.

Suprtool and STExport allow control over whether or not line feeds will be written to the output file or not. For details please see the Output Commands for both Suprtool and STExport.

Duplicate Output Files

If the output file already exists (and you haven't requested the Erase or Append option), Suprtool has to decide what to do.   This is how Suprtool/UX handles duplicate output files:

1. In Suprtool/UX, the duplicate output file processing takes place at the beginning of a task (in Suprtool/MPE it occurs at the end).

2. If Suprtool/UX is in batch, it purges any existing file with the same file name (Suprtool/MPE chooses a new output file name of the form OutputNN). If the Suprtool/UX task is on-line, it prompts the user to purge the file.

3. When Suprtool/UX purges a data file, it always deletes any associated .sd file, even if the output option is not Query or Link.

Classic Reals

Suprtool/UX does not support Classic real numbers (real or long).   If you are porting data files from MPE to HP-UX, you should first convert any Classic floating point numbers to their IEEE floating point equivalents.   You can do this by using the Extract command on Suprtool/MPE.

```
:run suprtool.pub.robelle
>base     sample
>get      customer
>define   ieee-credit-rating,1,4,ieee
>extract  cust-no
>extract  name-first
>extract  name-last
>extract  ieee-credit-rating = credit-rating
>out      mpefile
>xeq
```

The Classic and IEEE floating point formats are not identical.   Be sure to check the IEEE values after converting them from Classic floating point.

Input from Stdlist

In Suprtool/MPE, "input *" means read the input data from the stdin input device.   This is usually a job stream, and data is terminated by an !EOD symbol.

Suprtool/UX does not support reading data from stdin (via Input * or any other method).   If you need to create temporary data in the middle of a script, it is easy to use a temporary file.   For example, the following script creates a temporary file, writes three lines of data to it, then uses this file as input to Suprtool/UX.   At the end of the script we make sure that we remove the temporary file that we created.

```
#!/bin/sh

datafile=`mktemp`

echo "1234567 Line 1" >> $datafile
echo "2345678 Line 2" >> $datafile
echo "3456789 Line 3" >> $datafile

suprtool << !EOD
input    $datafile,rec 14,lf
define   key    ,1,7
define   line   ,8,7
extract  key
extract  line
list     standard
exit
!EOD

rm $datafile
```

Missing Features

The following Suprtool features on MPE are currently not available in Suprtool/UX:

¤ All IMAGE-related commands (Base, Get, Chain, Delete, Update, and Put). Use the Open and Select commands instead to access SQL databases.

¤ Edit command (Dbedit)

¤ Extracting a range of fields from an SQL database

¤ Export command (STExport exists as a separate program)

¤ Hints are not available

¤ Link command (Suprlink exists as a separate program)

¤ Table Command with the File option requires that the file being loaded is self-describing.

¤ Out= option of the Listredo command

¤ Output Ask, Num,Key, and Num,Query

¤ Output Temp (There are no temporary files in HP-UX)

¤ Output =Input

¤ Totals to a file

Suprtool Issues and Solutions

This section of the on-line help describes technical issues in using Suprtool. We have organized this material into several application areas:

TASK	Details about a Suprtool task.
ORACLE	How to perform Oracle extracts using Suprtool.
ALLBASE	How to perform Allbase extracts using Suprtool.
SD FILES	Self-describing files.
SORTING	How to sort quickly.
PERSONAL COMPUTERS	Creating files for PC applications.
POWERHOUSE	Quiz and other PowerHouse tools.
YEAR 2000	Year 2000 issues.
PERFORMANCE	Performance issues

A Suprtool Task

Suprtool's primary function is to extract data quickly; its focus is batch extraction. The key principle is: *the bigger the input data source and the smaller the subset of data selected, the more performance improves.*

Your aim is to replace serial reads and selection with Suprtool. To do this, break your task into components: an *input choice*, some *processing* selections, and the *output choice*.

Input Choices
Processing
Output Choices

Input Choices

Suprtool reads fixed-length data files.   You can create self-describing files with Suprtool's Query or Link output options.   It is easier to work with self-describing files because they have information about the fields in each input record.

Often you select a subset of the input records using the If command.   Only selected records are passed to the processing stage and the output choice.

Processing Selections

If you do not specify any processing, the input records are quickly copied to the output choice.   Some of your processing choices are

1. Sort the records into ascending or descending sequence (Sort or Key).   No records are output until all of the selected input records have been sorted.

2. Total one or more input record fields (Total).

3. Remove or select duplicate records (Duplicate).

Output Choices

Usually you wish to extract a subset of your records to feed into a report program, so the default output file is a data file.   The default output file format matches the input file format, unless you use the Extract command.   You can specify different formats for the output file by qualifying the Output command.   To have readable ASCII output, use "output xxx,ascii".   To produce "self-describing" files, use Output xxx,Link.

By default, every output record is identical to the corresponding input record.   The Extract command assembles output records by stringing together fields extracted from the input records.   With the Extract command you can insert constant values into the output record.

Each output record is written to the output choice.   You can also see a formatted listing of each record with the List command.

Large File Support

Suprtool can read, sort and write files greater than 2Gb on HP-UX 10.20 or higher.   Suprtool is limited to processing Large files with 2.1 billion records or less.

Suprtool and Allbase

Specify an Allbase database with the Open command. Once Suprtool has opened the database, use the Form command to obtain information about the tables in the database. Use the Select command to choose what data to read from your Allbase database.

We have tested the Allbase module with Allbase version G.1.09 for HP-UX. We believe that it will be compatible with future versions of Allbase. We have not tested Suprtool with any of the F versions of Allbase. Allbase access is available as a seperate add-on module to Suprtool.

Data-Types

When you specify a Select command, Suprtool figures out how to translate the Allbase internal data-types into ones that Suprtool can process. Not all Allbase data-types can be processed by Suprtool. The following table lists the Suprtool data-type that corresponds to the Allbase data-type:

Allbase Data-Type	**Suprtool Data-type**
Integer	Double
Smallint	Integer
Binary	Not Supported
Char	Byte
Varchar	Byte
Real	Ieee-32
Float	Ieee-64
Decimal	Packed
Numeric	Packed
TID	Not Supported
Date	Byte
Time	Byte
Datetime	Byte
Interval	Byte
Varbinary	Not Supported
Long binary	Not Supported
Long varbinary	Not Supported

Date and Time Types

Allbase has four types of fields that are associated with dates and times. These fields are converted to byte-type data and are returned with specific lengths.

The date and time fields are returned with the following byte lengths:

Data-Type	**Length**
Date	10
Time	8
Datetime	23
Interval	8

Restrictions

Suprtool cannot currently handle all Allbase database concepts.   The current restrictions are:

1. Suprtool requires that the ownername be specified when selecting a particular table in the following manner.

   ```
   >select * from purchdb.orders
   ```

   In this example the owner is purchdb and the tablename is orders.

2. Suprtool does not currently handle the Allbase date format.   You can convert the Allbase date format to something that Suprtool can handle with the TO_CHAR function in the Select statement.   For example:

   ```
   >select qty,TO_CHAR(date,'YYYYMMDD') from manufdb.testdata
   >def mydate,date[1],8          {redefine testdate }
   >item mydate,date,yyyymmdd     {define the date format}
   >if mydate<=$today(-900)
   ```

Suprtool and Oracle

You specify an Oracle database with the Open command.   You can open any Oracle database to which you normally have access.   If you cannot open your Oracle database, check with your system or database administrator so that your environment can be set up properly.   Once Suprtool has opened the database, use the Form command to obtain information about the tables in the database. Use the Select command to choose what data to read from your Oracle databases.
Oracle access is available as a separate add-on module to Suprtool.

Data-Types

When you specify a Select command, Suprtool figures out how to translate the Oracle internal data-types so that Suprtool can process them. Not all Oracle data-types can be processed by Suprtool. The following table lists the Oracle data-type and the corresponding Suprtool data-type:

Oracle Data-Type	**Suprtool Data-Type**
Varchar2	Byte
Number	Varies, see below
Long	Not supported
Rowid	Not supported
Date	Oracle Date
Raw	Not supported
Long raw	Not supported
Char	Byte
Mislabel	Not supported

<u>Number Data-Type</u>

Number Data-Type

Oracle numbers are translated into a variety of Suprtool data-types. The translation depends on the precision of the number and the number of decimal places. The following table describes the translation for each case:

Precision	**Decimal Places**	**Suprtool Data-Type**
None	Any	8-byte IEEE
1-4	Zero	2-byte Integer
5-9	Zero	4-byte Integer
1-9	Non-zero	Packed-decimal
10-27	Any	Packed-decimal
28-38	Any	8-byte IEEE

In packed-decimal translations Suprtool uses the precision of the number to determine the size of a packed-decimal number.

Order By vs. Sort

Oracle's Order By statement on the Select command does not always generate the same results.   Specifically, sorted fields with null field values appear at the beginning when they are sorted by Suprtool's Sort command.

Restrictions

Suprtool/UX cannot handle all Oracle database concepts.   The current restrictions are:

1. Suprtool/UX can handle varchar2, char, date, and number data-types.   It cannot handle any other data-type.

2. Because any Oracle Select command can be used, it is possible to generate column names that are not compatible with Suprtool/UX.   For example,

```
        >select sal + comm from bonus
```

This example produces a column called "sal + comm".   In some cases Suprtool/UX correctly uses this as the column name (e.g., the List command).   You cannot refer to this column by name in any Suprtool/UX command that accepts field names as a parameter.

SDUnix Utility

SDUnix is an MPE program that takes self-describing file information and writes it out to an MPE flat file.   This flat file can then be transferred to HP-UX together with the data file so that Suprtool/UX can reference the self-describing information about the fields.

To copy the SD file to the HP-UX machine, it must have an .sd extension and be in the same directory as the data file.   For example, if the data filename is /usr/local/data/datafile, the SD file must have the name /usr/local/data /datafile.sd.

The SD file is written out to the same domain (permanent or temporary) as the input file. The SD file contains only one record with the necessary length to store all of the label information.

Installation

SDUnix has been included on your HP-UX tape and needs to be installed on your MPE system.   You can do this with either FTP or DSCOPY.

Installing using FTP

1) Log on as Mgr.Robelle

```
:hello mgr.robelle,pub
```

2) Remove the Sdunixnm file

```
:purge sdunixnm.pub
```

3) FTP the file from your Unix machine

```
 ftp dopey
 binary
 cd /opt/robelle/mpe
 get sdunixnm sdunixnm;rec=128,1,f,binary;code=nmprg;disc=1400
```

4) Remove any old versions of Sdunix

```
:purge sdunix
```

5) Rename the program file

```
:rename sdunixnm,sdunix
```

To install the CM version of SDUnix, specify Sdunixcm as the file name, ;disc=400,1,1 for the file size, and PROG for the filecode.

Installing with DSCOPY

1) Log on as Mgr.Robelle

```
:hello mgr.robelle,pub
```

2) Remove the Sdunixnm file

```
:purge sdunixnm.pub
```

3) Specify file command for proper program file size

```
:file sdunixnm;disc=1400
```

4) DSCOPY the file from your Unix machine

```
 dscopy /opt/robelle/mpe/sdunixnm:hpux[user:password] to
  *sdunixnm;bin;fcode=1030;rsize=256;fix;rep
```

5) Remove any old versions of Sdunix

```
:purge sdunix
```

6) Rename the program file

```
:rename sdunixnm,sdunix
```

To install the CM version of SDUnix, specify Sdunixcm as the file name, ;disc=400,1,1 for the file size, and 1029 for the filecode.

SDUnix Parameters

All SDUnix parameters are specified via Info = *string*.   There are three parameters:

input-file   *sd-file*   LF | NOLF

The first parameter is the name of an MPE self-describing file.   The second parameter is the name of the .sd file that SDUnix created.

LF vs. NOLF

Use the third parameter to specify whether the data file has LF (line feed) as the record separator, or whether the file does not use a file separator.   If you use FTP to copy the data file to your HP-UX machine, you should specify the NOLF option and be sure to use a binary mode transfer.   If you are using DSCOPY (with its default options) to copy the data file, you should specify the LF option.

Examples

The following section contains examples of creating an SD file on MPE, converting the SD information, and finally copying the two files to an HP-UX machine.

First create an SD file with

```
:run suprtool.pub.robelle
>base store,5
Database password [;]?

>get d-inventory
>out dinv,link
>exit
```

Now you can convert the label information to an .sd file using the SDUnix utility.   Note that the data file is the first file passed in the info string.
Specify LF if you are using DSCOPY.

```
:run sdunix.pub.robelle;info="dinv dinvsd lf"
```

Now you can use DSCOPY to copy the files to the HP-UX machine.

```
:dscopy dinv   to store.dinv   :dopey[data:password]
:dscopy dinvsd to store.dinv.sd:dopey[data:password]
```

Specify NOLF if you are using FTP.

```
:run sdunix.pub.robelle;info="dinv dinvsd nolf"
```

Use FTP to copy the files to the HP-UX machine.

```
:ftp dopey
ftp> binary
ftp> exitonerror
ftp> put dinv /users/data/store.dinv
ftp> put dinvsd /users/data/store.dinv.sd
ftp> quit
```

Now you can use Suprtool/UX to read the SD file.

```
$ suprtool
>input store.dinv
>form
   File: store.dinv     (SD Version B.00.00)  No line feeds
      Entry:                     Offset
         BIN-NO               I1      1
         LAST-SHIP-DATE       I2      3
         ON-HAND-QTY          I2      7
         PRODUCT-NO           Z8     11
         SUPPLIER-NO          Z8     19
         UNIT-COST            P8     27
         ITEM-DESC1           X20    31
         ITEM-DESC2           X20    51
         ITEM-DESC3           X20    71
         ITEM-DESC4           X20    91
   Entry Length: 110  Blocking: 1
>out dinvfile
>xeq
```

Link vs. Query

SDUnix and Suprtool/UX can recognize files created with the ,Query option and from Query. However, they cannot recognize compound item details or any Item attributes, such as Decimal or Date type.

Suprtool and Self-Describing Files

A problem with data files is that there is no field information. Self-describing files solve this problem by providing field information about the file.   Suprtool reads and writes SD files; Suprlink requires SD files as input and creates an SD file as output.

Create an SD File from a Table

You request an SD file using the Link option of the Output command.   If you extract columns from the table, only the extracted columns appear in the SD file.

```
     >select * from sales          {input from a table}
     >output salefile,link         {salefile has all of the}
     >xeq                          {    fields from sales}
```

Create an SD File from a Data File

You must Define and Extract the fields you want to have in the SD file. Use the Link option of the Output command to create the file as a self-describing file. Although Suprtool itself allows longer field names, SD files only store the first 16 characters of a field name.

```
>input sales.data,reclen 16,nolf  {a data file}
>define cust_no,1,6,byte
>define invoice_date,7,6,integer
>define sales_qty,13,4,packed
>extract cust_no,invoice_date,sales_qty
>output salefile,link             {salefile has all of the}
>xeq                              {    extracted fields}
```

SD Files as Input

When you specify an SD file as input to Suprtool, all the field information becomes available. You can select, extract, and total fields without the Define command.   In addition, the Input command no longer needs any Reclen or LF parameters.

```
>input salefile         {self-describing file}
>form                   {display the fields in the file}
>if sales_total > 10000 {select based on a field}
>extract cust_account   {only extract a few fields}
>extract sales_qty
>extract sales_tax
>extract sales_total
>total sales_total      {total a field from the file}
>output newfile,link    {create a new SD file}
>xeq
```

Listing SD Files

Suprtool normally lists data files in an Octal/Char format.   When listing an SD file, Suprtool produces a formatted listing with field names and field values converted into ASCII:

```
>input salefile        {self-describing file}
>list                  {produce a formatted listing}
>xeq
```

Decimal Places and Date Formats

You use the Item command to identify items with an implied number of decimal places or a date format. If you create a self-describing file, this information is retained. When you input such a file, all Suprtool commands are automatically informed about the decimal places and date formats. The Form command shows these extra attributes as comments at the end of each field description. For example,

```
>input salefile         {self-describing file}
>item deliv_date ,date   ,yyyymmdd
>item purch_date ,date   ,yyyymmdd
>item sales_tax  ,decimal,2
>item sales_total,decimal,2
>output newfile,link    {creates SD file with item attributes}
>xeq
>form   newfile         {shows decimal points and dates}
```

Restrictions of SD Files

So far in this section, we have shown how to create self-describing files using the Link option of the Output command. The Link option produces a special form of self-describing file. Not all software can read this form of self-describing file. You can use the Query option to create an old-style self-describing file. The Query option has the following restrictions.

Self-describing files were originally created by HP in MPE so that files could be fed into HPWORD and HP graphics packages. One problem with HP's definition was that no provision was made for compound fields (e.g., 10J2). When Suprtool creates an SD file with compound fields via the Query option, it uses a special data-type. When you input such a file to Suprtool, all compound fields are treated as byte arrays. Suprtool correctly copies and extracts these fields, but you can not select with them. The Query option is not capable of retaining information about decimal places or date formats.

Suprtool and Sorting Files

When Suprtool sorts two records that have the same key value, the first record read by Suprtool is the first record on the output file.   For data files, this means that input records with the same key values appear in the same order in the output file.

Suprtool and Personal Computers

You can format files to be downloaded to your PC for use in spreadsheets or databases with the PRN option of the Output command.   Suprtool formats your file as a PC structure (a comma-delimited file).   Not all PC applications support the PRN format.   For more precise data conversion, create a self-describing file then use STExport.   See the STExport manual for details. You transfer the Suprtool output file to your PC and then import it into your PC application.

Downloading to the PC

After you have created a PRN file using Suprtool, you can use FTP or any of the many terminal emulator programs available to download the file to the PC. This includes Reflection from Walker Richer & Quinn.

Decimal Places

Be sure to specify which fields have decimal places when creating the PRN file. Suprtool reserves extra space for decimal points that appear in the PRN output. When formatting numeric fields, Suprtool inserts the decimal point at the correct place. When you import your file into your PC application, numeric fields are automatically formatted correctly.

Spreadsheets

The following procedure allows you to include literal headings in your spreadsheet using only one file, the size and shape of which is computed by STExport. We have tested this method with MS Excel; it should work with any spreadsheet that supports the importing of delimited files.

There are two steps. First, build a self-describing file with Suprtool, then use STExport to convert it to PRN and add the headings.

1
2

1. In Suprtool build a self-describing file:

```
>input   ...
>define  items...
>item    items...
>extract fields...
>output  sdfile,link
>xeq
```

2. In STExport convert to PRN and add the header line:

```
     $input   sdfile
     $heading fieldnames
     $output  pcdata
     $exit
```

The file Pcdata is a variable-length PRN file with both headings and data.

Paradox Databases

Paradox is a PC database product from Borland International. Paradox imports PRN files in one of two ways. Here are the advantages and disadvantages of each method:

1. Import the file into an existing table. This table must have been set up with all the field names, field lengths, and field types. As long as the corporate data structure doesn't change, this method works fine. It is less work than the second method, since you only have to set up the table structure once.

2. Import the file into a new table. Paradox reads the input file twice. The first time it determines the maximum length of each field. The second time it loads the data into the table. After the file is loaded, you must define the field names and possibly the field types. Fortunately, you can automate this part of the process with a script. While loading takes longer, each field is allocated enough space for the largest value, saving disc space.

We would recommend the second method if you are making a read-only copy of corporate data. This method helps protect the Paradox application from some changes in the corporate application (e.g., an expanded field length). The first method is needed if you want to change the data, since you need the ability to extend data fields.

Suprtool and PowerHouse Applications

You can use Suprtool to significantly speed serial extracts using Quiz and QTP from Cognos. In many cases the changes to existing applications are minor.

Suprtool with Quiz/QTP

Quiz, QTP, and Quick are components of PowerHouse 4GL, a popular fourth generation language sold by Cognos. You can improve the performance of Quiz reports by using Suprtool to select and sort the records from a file or SQL table, and passing selected records to Quiz for final reporting. To do this, you need some way to tell Quiz about the record structure of Suprtool's output file. Quiz already has the capacity to do this without making any changes to the PowerHouse PDL dictionary.

In the following example, Suprtool extracts records from the Custmast file, sorts them, and writes them to the Cmasfile file. These are the records we need for the Quiz report.

Step 1: Create a Subfile with Quiz

The first step is to use Quiz to extract one entry from the Custmast file and write it to a PowerHouse "subfile". Note that you could also use QTP to build the subfile.

```
     $rm cmasfile.sf
     $rm cmasfile.sfd
     $quiz
     >access custmast
     >report summary all
     >set subfile name cmasfile keep
     >set report limit 1
     >go
```

This creates 2 files: Cmasfile.sf, containing the extracted data, and Cmasfile.sfd, containing the PowerHouse record definition for the data portion of the subfile. We use Suprtool to reload the Cmasfile.sf file with all the records required for the final report.

Step 2:   Output Erase in Suprtool

Once you have created the PowerHouse subfile, use the Erase option of the Output command in Suprtool to load the file.   This overwrites any data in the data portion of the subfile, but it does not affect the "dictionary" for that file.

```
     $suprtool
     >input custmast,reclen 80,nolf
     >if credit_limit>=1000000
     >sort custnum
     >output cmasfile.sd,erase
     >exit
```

Step 3: Report with Quiz

The Cmasfile.sf file now contains the sorted records for the Quiz report. Quiz can access the file because the Cmasfile.sfd contains the record definition. You can now use Quiz to generate the report:

```
$quiz
>access *cmasfile
>report ...
>go
```

Using QTP to Create Subfiles

You can also use QTP instead of Quiz to create the PowerHouse subfile:

```
$rm cmasfile.sf
$rm cmasfile.sfd
$qtp
>access custmast
>set input limit 0
>subfile cmasfile keep include custmast
>go
```

Creating Subfile with Script File

The process of creating a subfile is essentially the same regardless of the input file being used.   The only things that change are the subfile and input file names.   Here is a script file for automating this task.   This script can be executed before running Suprtool or from within Suprtool.

Makesub

```
#!/bin/sh
rm usefile
echo set report limit 1                              >usefile
echo access $2                                       >>usefile
echo set subfile name $1 keep                        >>usefile
echo report summary all                              >>usefile
echo go                                              >>usefile
echo exit                                            >>usefile
quiz                                                 <usefile
```

This script accepts two parameters: the name of the subfile to be created and the name of the file whose structure it will duplicate.

Makesub can be invoked from within Suprtool, as shown below:

```
$suprtool
>:makesub cmasfile, custmast
>input custmast,reclen 80,nolf
>if credit_limit>=1000000
>sort custnum
>output cmasfile.sf,erase
>exit
```

Quiz: Generating Suprtool Commands

Suprtool does not have the ability to prompt users for selection criteria. You can easily create a short Quiz procedure to prompt for values and create a file of Suprtool commands.

```
> define D-PROMPT date format YYMMDD =            &
>    parm prompt "Enter selection date: "
> report &
>    "input custmast,reclen 80,nolf"              &
>    skip "if DATEFIELD = " D-PROMPT pic "^^^^^^" &
>    skip "output OUTFILE"    &
>    skip "xeq"
> set nohead
> set report device disc name STCODE
> set page length 0
> go
```

Quiz edits the user's input for a valid date.   Note that there is no ACCESS statement, but Quiz still writes one record to the output file.   Execute the resulting commands as a Suprtool usefile:

```
$suprtool < STCODE
```

Year 2000 Solutions with Suprtool

Suprtool often has to process dates in both the twentieth and twenty-first centuries. If you include the century in your dates, Suprtool should behave as most users expect. If you do not include the century in your dates, how Suprtool behaves will be dependent on your specific application and data.

What If I Have Four-Digit Years?

If your dates have four-digit years, Suprtool should work as expected. Selection based on the $today or $date features will select dates in both the twentieth and twenty-first centuries. Dates that do not collate correctly (e.g., mmddccyy) will not be accepted by Suprtool's If command for relative selection (e.g., <, <=, >, or >=). If you have these date formats you can use the $stddate function, converts any date format to a ccyymmdd type date.

Suprtool, as it has always done, will continue to sort dates based on their numeric value and not on any implied date order.

In Suprtool 4.0, we introduced some new command parsing features that let you control how Suprtool parses the year of the $Date function. You can either use two-digit years by applying a cutoff rule or you can force all years to be specified as four digits.

Set Date Cutoff
Set Date ForceCentury
Two-digit years
aammdd Dates
Invalid Dates
Converting Dates

What does Set Date CUTOFF do?

Date Cutoff tells Suprtool what century to use when Suprtool generates the constant date value from the $date function.

Before version 4.0, Suprtool would assume 19 for the century for any user-specified $date with a two-digit year.   For example:

```
>item date-field,date,ccyymmdd
>if date-field <= $date(40/12/26)
```

Previously the $date function would convert the user specified $date to 1940/12/26 in order for it to be compared to the date-field format of CCYYMMDD.   Now with Set Date Cutoff xx,

Suprtool assumes 20 for the century if the two-digit year specified in the $date function is less than the value of Set Date Cutoff.   For example,

```
>set date cutoff 50
>item date-field,date,ccyymmdd
>if date-field lt;= $date(40/12/26)
```

Suprtool in this case assumes the full $date to be: 2040/12/26

The default value of Set Date Cutoff is 10.

Stddate and Set Date Cutoff

When $Stddate has to convert from a date with only a two-digit date, the conversion to the four-digit date will use the value of Set Date Cutoff when converting the date.

For example,

```
> in /data/sales/ordfile
> set date cutoff 15
> def new-ship-date,1,4,double
> item ship-date,date,mmddyy
> ext order-no / sales-amount
> ext new-ship-date = $stddate(ship-date)
> out salesinfo,link
> xeq
```

If any ship date is 15 or below then the century applied to the new-ship-date field with a year value of 15 or less will have a century of 20 applied.

What does Set Date FORCECENTURY do?

Set Date ForceCentury On will not allow a yy date to be entered in the $date function, it will force the user to enter a full ccyy date.

```
    >set date forcecentury on
    >item date-field,date,ccyymmdd
    >if date-field >= $date(98/12/10)

  Error: You must specify the century or Set Date ForceCentury off
```

The default value for Set Date ForceCentury is off.

What If I Have Two-Digit Years?

If you have dates with two-digit years, there are two main solutions to making your application ready for the Year 2000:

1. Convert all of your date data to use four-digit years and modify your programs to process four-digit years, or

2. Assume that certain dates are in the twentieth century and some in the twenty-first (this is usually called date windowing).

The first solution requires that you change all Suprtool Item commands for two-digit years to a four-digit year format. If you have not already done so, you may want to isolate all of these Item commands in a single file per input source (e.g.,one use file for every database or file)

You may also want to use Suprtool to assist you in changing your actual data from two-digit years to four. See the Can Suprtool convert Two-Digit Years to Four-Digits? section of this manual for more details.

If you do not include the century in your dates (the second solution above), you will have the following problems:

1. Selecting dates in YYMMDD format will not produce the expected results in relative operations (e.g., <, <=, >, or >=). You will need to change all of your If commands to use the $stddate function.

2. Sorting dates that include both 20th and 21st century dates, will not collate the way most users expect, whether with Suprtool, COBOL sort verbs, or sort tools. These tools sort based on the numeric value of a date. To make this work correctly within Suprtool, you will need to use the $stddate function in an Extract command to generate a date with a four-digit year, then sort on this new date field with another Suprtool task.

What Is Wrong with Two-Digit Years?

Currently the date format of yymmdd collates (sorts) correctly if the date is not beyond December 31, 1999. Given the current date is 981210, numerically this is less than next year whose date value is 991210.

At the turn of the century dates in the yymmdd format (or yymm) will no longer sort correctly because the value of December 10, 2000 (001210) is less than 981210.

Consequently, if we have a date beyond 1999, stored in YYMMDD format, a relative operation such as:

```
    >if date-field >= $date(98/12/10)
```

will not find the date of December 10, 2000. You will need to use the $stddate function to make this task work correctly.

```
>if $stddate(date-field) >= $date(98/12/10)
```

How Do $Today and $Date Work?

Suprtool's date functions ($date and $today) are a short-hand method of generating a numeric constant. So a date selection like:

```
>item invoice-date,date,YYMMDD
>if invoice-date < $today
```

is exactly the same as

```
>if invoice-date < 980401        {on 1st April, 1998}
```

Suprtool does record selection on the *numeric* value of the field and not on the implied date value. If we move the calendar ahead to January 1, 2000 and do the same commands as above, the result would the same as if you had typed:

```
>if invoice-date < 000101        {on 1st January, 2000}
```

If you have some invoice dates from the previous century (e.g., 990101 for December 1st, 1999), they will not be selected.

Will Suprtool Generate an Error for Two-Digit Year Dates?

Sometimes.

Because dates beyond 1999 will not collate properly for the YYMMDD and YYMM formats, starting in version 4.0.11 the If command produces an error if the year specified in a $date or $today function is greater than 1999 and the date format is yymmdd or yymm, and you are performing a relative operation (e.g., <, <=, >, or >=).

```
>item enddate, date, yymmdd
>if   enddate >= $date(*+4/*/*)        {21st century date}
                        ^
Error:  Cannot use a date beyond 1999 for this format
```

Suprtool returns this error by default, but you can override it with the following set command:

```
>set date ifyy2000error off
```

This tells Suprtool to allow the previously described relative operations and suppress the error message. While you can override the error checking, the behavior of $today and $date is not changed.

How Do I Use $Today and $Date with yymmdd Dates?

If you need to have Suprtool select dates in yymmdd format with $today or $date, you need to use one of the following solutions:

1. Change the date storage format to include the century in all datasets and data files, so you can use the following item command:

   ```
   >item invoice-date,date,CCYYMMDD
   ```

2. Use the $stddate function that adds the century component to dates in a ccyymmdd format in a J2 container.

See the sections:

How do I convert a J2 date from yymmdd to ccyymmdd? and How do I convert an X6 yymmdd date to an X8 ccyymmdd date?

for more specific details on converting two-digit year date formats into four-digit year date formats.

aammdd Date Format

The aammdd date format was developed by James Overman of HP for use in their MM3000 product.   This format is only available for the X6 data-type.

The aammdd format is similar to yymmdd, but the year portion of the date uses a combination of numbers and letters of the alphabet to represent years beyond 1999.

By substituting a letter of the alphabet in the first position of the year, we can extend a six-digit date and also ensure that the dates collate correctly. For example,

YY of AAMMDD	CCYY
A0 - A9	2000 - 2009
B0 - B9	2010 - 2019
C0 - C9	2020 - 2029

Because letters are greater than numbers in the collating sequence we can ensure that aammdd dates beyond 1999 will order correctly.

Suprtool also supports other date formats with this two-digit year representation.   These formats are aamm, mmddaa and ddmmaa.

Invalid Dates

The If command has a $invalid function to find all invalid dates for a particular field.   An invalid date is any number of a particular date format whose date equivalent cannot be found on the calendar.   For example, a date with a month of 99 will be considered invalid.

```
>in /data/salesfile
>item deliv-date,date,ccyymmdd
>if $invalid(deliv-date)
>out baddates,link
>xeq
```

Can Suprtool Convert Two-Digit Years to Four-Digits?

Suprtool is capable of converting dates from one format to another using a variety of Suprtool features. We will show how Suprtool can convert common dates without the century to those that have the century included. While Suprtool can convert your data, it is up to you to change your programs. Adager a third-party program for changing Image database structures has the ability to change date fields. Suprtool can convert data in Oracle and Allbase databases, flat files and self-describing files.

J2 YYMMDD
X6 YYMMDD
X6 MMDDYY

Case 1:   Converting a J2 Date from yymmdd to ccyymmdd

The $stddate function can convert six-digit date formats to ccyymmdd.   But what if all the dates are not actually dates, but some dates are filled with 9s as a flag to an application?

Here is a data file with two date fields, which are J2 items in the date format yymmdd.

```
    File: kb.calls     (SD Version B.00.00)  Has line feeds
       Entry:                     Offset
          CALL-NBR              I2      1
          CALL-TAKER            X20     5
          CALL-CATEGORY         X2     25
          STATUS-FLAG           X2     27
          COMPANY-NAME          X60    29
          SERIAL-NBR            X4     89
          DATE-CREATED          J2     93
          DATE-MODIFIED         J2     97
    Entry Length: 100  Blocking: 1
```

First, you need to know and understand your data.   Are there invalid dates? If so, does the value have some other logical meaning?   Are they, for example, flags for your application?

```
>in kb.calls
>item date-created,date,yymmdd
>item date-modified,date,yymmdd
>if $invalid(date-created) or $invalid(date-modified)
>list
>xeq
```

After fixing up the dates that are incorrect you can now get ready to start converting the dates.

This can be done in one step with the $stddate command.

```
>in kb.calls
>item date-created,date,yymmdd
>item date-modified,date,yymmdd
>if not $invalid(date-created) and not $invalid(deliv-date)
>ext call-nbr / serial-nbr
>ext date-created  = $stddate(purch-date)
>ext date-modified = $stddate(date-modified)
>out kb.calls.new
>xeq
```

You can now replace the old kb.calls file (and associated self-describing file) with the new kb.calls.new file.

Case 2:   X6 yymmdd Data to X8 ccyymmdd

The following Suprtool task shows how you can generate a new file.

Consider the following self-describing file and its date-created and date-modified fields:

```
    File: kb.calls     (SD Version B.00.00)  Has line feeds
       Entry:                     Offset
          CALL-NBR             I2      1
          CALL-TAKER           X20     5
          CALL-CATEGORY        X2     25
          STATUS-FLAG          X2     27
          COMPANY-NAME         X60    29
          SERIAL-NBR           X4     89
          DATE-CREATED         X6     93
          DATE-MODIFIED        X6     99
    Entry Length: 104  Blocking: 1
```

You want to convert to a date format with room for a cc at the beginning of the date.   In order to convert these dates you need to be able to put either a 19 or 20 in front of the yymmdd date, depending on the value of the year. Before you can do either of these you must confirm, once again, that you have no invalid dates.

```
     >in kb.calls
     >item date-created,date,yymmdd
     >item date-modified,date,yymmdd
     >if $invalid(date-created) or $invalid(date-modified)
     >list
     >xeq
```

Once you have confirmed that there are no invalid dates you can start converting the dates that you have.   Because there are two date fields in this file you must be careful to add the appropriate century for the proper field. For this example, assume that if a year is less than 1950 then the century should be 20.

```
     >in kb.calls
     >item date-created,date,yymmdd
     >if date-created >= $date(1950/01/01)
     >out kb.calls.new
     >ext call-nbr / serial-nbr
     >ext "19"
     >ext date-created
     >ext date-modified
     >xeq
```

Now insert 20 in the century for the appropriate records:

```
     >in kb.calls
     >if deliv-date < $date(1950/01/01)
     >ext call-nbr / serial-nbr
     >ext "20"
     >ext date-created
     >ext date-modified
     >out kb.calls.new,append
     >xeq
```

Now you can convert the other field from the flat file, and add a century to the date-modified field:

```
>reset
>in kb.calls.new
>item date-modified,date,yymmdd
>if date-modified >= $date(1950/01/01)
>out kb.calls.new2,link
>ext call-nbr / date-created
>ext "19"
>ext date-modified
>xeq
IN=19716, OUT=19716.  CPU-Sec=5.  Wall-Sec=9.
```

Because you extracted all 19716 records you know that you do not have any records with the purch-date field that need to be updated with a 20.

Case 3:   X6 mmddyy Data to X6 yymmdd

The following Suprtool task shows you how to convert a date in a self-describing file from mmddyy to yymmdd format.

Consider the following self-describing file and its date-created and date-modified fields:

```
     File: calls     (SD Version B.00.00)  Has line feeds
       Entry:                     Offset
          CALL-NBR             I2      1
          CALL-TAKER           X20     5
          CALL-CATEGORY        X2     25
          STATUS-FLAG          X2     27
          COMPANY-NAME         X60    29
          SERIAL-NBR           X4     89
          DATE-CREATED         X6     93        <<MMDDYY>>
          DATE-MODIFIED        X6     99        <<MMDDYY>>
    Entry Length: 104  Blocking: 1
```

You want to convert these two dates to a data format of yymmdd before adding a century in front of the year.   This can be easily accomplished by defining each sub part of the date and extracting them in the new order.

```
   >in calls
   >def date-created-mm,date-created[1],2
   >def date-created-dd,date-created[3],2
   >def date-created-yy,date-created[5],2
   >def date-modified-mm,date-modified[1],2
   >def date-modified-dd,date-modified[3],2
   >def date-modified-yy,date-modified[5],2
   >ext call-nbr / serial-nbr
   >ext date-created-yy
   >ext date-created-mm
   >ext date-created-dd
   >ext date-modified-yy
   >ext date-modified-mm
   >ext date-modified-dd
   >out calls2,link
   >xeq
```

You now have a file with the dates in yymmdd order, but the self-describing information shows three separate fields.

```
    File: calls2     (SD Version B.00.00)  Has line feeds
       Entry:                     Offset
          CALL-NBR             I2      1
          CALL-TAKER           X20     5
          CALL-CATEGORY        X2     25
          STATUS-FLAG          X2     27
          COMPANY-NAME         X60    29
          SERIAL-NBR           X4     89
          DATE-CREATED-YY      X2     93
          DATE-CREATED-MM      X2     95
          DATE-CREATED-DD      X2     97
          DATE-MODIFIED-YY     X2     99
```

```
            DATE-MODIFIED-MM     X2    101
            DATE-MODIFIED-DD     X2    103
      Entry Length: 104  Blocking: 1
```

You can convert these several fields to one field with another extract task:

```
     >in calls2
     >def date-created,93,6,byte
     >def date-modified,99,6,byte
     >item date-created,date,yymmdd
     >item date-modified,date,yymmdd
     >ext call-nbr / serial-nbr
     >ext date-created
     >ext date-modified
     >out calls3,link
     >xeq
     IN=19716, OUT=19716.  CPU-Sec=5.  Wall-Sec=9.
```

You now end up with a file that looks like this:

```
      File: calls3     (SD Version B.00.00)  Has line feeds
         Entry:                     Offset
            CALL-NBR             I2      1
            CALL-TAKER           X20     5
            CALL-CATEGORY        X2     25
            STATUS-FLAG          X2     27
            COMPANY-NAME         X60    29
            SERIAL-NBR           X4     89
            DATE-CREATED         X6     93      <<YYMMDD>>
            DATE-MODIFIED        X6     99      <<YYMMDD>>
      Entry Length: 104  Blocking: 1
```

You then add the century to these fields as described above.

Performance Issues

HP-UX sites use Suprtool because it provides access to their data many times faster than they are used to. Suprtool also enables them to perform time-consuming DP functions with only a few simple commands. The typical Suprtool task consists of extracting some data for a report, then feeding the Suprtool output file into the final report program. For example, you might fill a data file with the subset of data needed, then report that file with Microfocus COBOL.

Sort Performance

Suprtool/UX uses its own set of sorting routines.   These routines are generally faster than the sorting algorithms provided with software tools and SQL databases.

If you have the right data, Suprtool/UX can sort much faster than other HP-UX tools. Because sort performance varies a lot from application to application, we recommend that you test Suprtool/UX in your own environment.

You can improve the performance of sort operations by moving the sort workspace to a different physical disk drive than the input file uses.

You can move the sort scratch space by setting the TMPDIR environment variable to a directory that resides on another physical disk drive, provided you have read and write access to that directory.

```
     export TMPDIR="/var/tmp/sortscratch"
```

Oracle Performance

Suprtool/UX provides you with easy ways to let either Oracle or Suprtool do most of the work. Whether it is best to use Oracle or Suprtool depends on your specific machine, database, and application. You can use the Select command to force Oracle to do much of the processing or you can use Suprtool to do the work. In our testing, Suprtool consistently sorts two to four times faster than Oracle. Your performance improvements may be different from ours, so we recommend that you take some common tasks and try them with both tools. Here is an example of sorting with Oracle and then with Suprtool:

Oracle sorts data:

```
>select * from emp order by ename
```

Suprtool sorts data:

```
>select * from emp
>sort ename
```

For more information on Oracle performance, we recommend the book *Oracle Performance Tuning* by Peter Corrigan and Mark Gurry (published by O'Reilly and Associates).

Analyzing Performance Data

It is better to test Suprtool with your own database and your own application needs, rather than trust a "generic" performance test.   The ideal test is an actual production report whose bad performance is causing you a problem.   If you obtain improvements by using Suprtool, you know that you can achieve better speed in practice as well as in tests.

Use Suprtool as a front end to your problem report, producing a small extract file that contains just the fields and records needed for your final program. Once you get that working, consider linking in data from other files or datasets using Suprlink.   For comparison purposes, run the Suprtool test at the same time as you would normally run the original program.   Comparing a standalone midnight run against a mid-day run does not give valid results, nor does comparing two runs in succession (the second run benefits on HP-UX because the files are already in memory at that time).

Performance Summary

Suprtool *is* a performance tool. Not because every Suprtool command is always fast, but because performance is one of our first concerns when adding new features to Suprtool. We always look for the fastest way to do any task, even if that means it can't be quite as flexible as other software tools.

Suprtool Commands

When you run Suprtool, it prompts for commands with a ">" character and reads command lines from the standard input device.   Suprtool commands contain a command name which may include one or more optional parameters that are each separated by commas.

In this chapter, we describe the Suprtool commands in alphabetic order.   Each command name is followed by its minimal abbreviation in brackets.   For example: [D] for Define and [DU] for Duplicate.

Most Suprtool commands work within the context of the input file.   In general, the Open, Select and Input commands must be entered before other commands. Once the input source has been specified, commands can be entered in any order.

Abbreviating

You may shorten the command name to any substring that uniquely defines the command. For example, Form can be abbreviated as FO or F, since there are no other commands that start with "F".   Duplicate, however, can be abbreviated only to DU, since there is also a Define command in Suprtool.

```
>i sdfile               {Input command}
>l                      {List command}
>x                      {Xeq command}
```

Uppercase or Lowercase

You can enter the letters in either uppercase or lowercase because Suprtool upshifts everything in the command line except literal strings within quotes (e.g., "abc") and disc file names.   These two commands are identical:

```
>EXTRACT QTY
>extract qty
```

Multiple Commands per Line

You can enter several commands on a single line, if you separate them with semicolons. An entire "task" can be placed on one input line.

```
>in sdfile;out new;xeq
>in sdfile;if cust_status<>10,20,30;list;x
```

Caution: Suprtool cannot distinguish between several commands on one line and several commands entered on several lines. This is not a problem when using Suprtool in batch, as Suprtool stops executing when an error occurs. But when using Suprtool interactively, specifying multiple commands separated by a semicolon, Suprtool keeps on parsing the rest of the line after it finds an error. For example, if you misspell the "fldname" when you type the following,

```
>in  sdfile; if fldname="value";out filename;xeq
```

Suprtool sends you an error message with the typo, but continues with the rest of the command line. This has the effect of selecting all the entries in the datafile. It is risky to type Xeq on the same command line as any other command unless:

1. You are not concerned with the consequences if you make a mistake (e.g., any "extract" task should be safe).

2. You don't make any mistakes.

The usual reason for putting all the commands on one line, including the Xeq, is to permit repetition of the task by using the Before command. It is not necessary to type everything on one line because with the Listredo command, Suprtool allows you to pick and choose any of your last 1000 commands.

Continuation

The maximum *physical* command line is 256 characters. You may enter commands on multiple input lines by putting an "&" continuation character at the end of the line.   The maximum total command length is 256 characters.   The separating comma in commands is not optional.   Should your If command exceed 256 characters, use the Table command, or $read.

```
>in sdfile
>if status="20" and &          {continue the If command}
    state="AZ","CA","OR"       {select several states}
>output outfile
```

Comments on Command Lines

Comments may appear at the end of any command line, as long as they are surrounded by curly braces. Many of the examples in this manual show comments at the end of command lines. You can enter a comment as the only item in a Suprtool command line. When you enter continued command lines, the comment can appear before or after the continuation character:

```
>{The following task extracts all customer records for}
>{    the different customers we are interested in.}
>in customer                   {input self-describing file}
>if status = "10" or &         {prepaid status}
    status = "20" or &         {current status}
    status = "30"              {arrears status}
>output outfile                {output to a disc file}
>exit                          {execute the task and exit}
```

HP-UX Commands

If Suprtool doesn't recognize the command you have entered, it tries to interpret it as an operating system command.   Suprtool also interprets any command line beginning with an exclamation mark (!) or a colon (:) as an O/S command.   For example:

```
     >!     sort custfile by custnum  {comment}
     >input custfile
     >key 1,10                        {no '!' on the next command}
     >ls sort*                        {check for file name}
     >out  sortcust
     >exit
```

Calculator

Any command line beginning with an equal sign (=) is treated as a calculator expression. This feature can be used to compute blocking factors and do other calculations without the need of an electronic calculator.   For help, type =?.

Control-Y Interrupt

You can interrupt a Suprtool task with the Control-Y key (hold down Control while striking Y). Suprtool responds by telling you how much work it has done (IN=,OUT=,etc.) and asks if you wish to stop. Hit the Return key to continue or type YES to stop the task. Even though you abort the task, your output file is saved (although it may be empty if you stop before the sort phase is over).

Many HP-UX sites use Control-C as the interrupt key instead of Control-Y. Use the HP-UX 'stty' command to display your 'intr' setting.

Add Command   [Add]

Specify an SQL table to which you wish to Add records.

    Add *tablename*

Use the Add command to "insert" records into an Oracle table.   You must specify the Oracle *tablename* and you must have opened the Oracle database to which you wish to add records.

The tablename specified must be a valid table and not a view.   The fields from the input source, or the extracted names must be the same as the column names in the table to which you wish to add records.

You do not have to specify all columns in the table; unreferenced columns will be given default values depending on their data-type.

You cannot currently add records from another SQL table.   However, you can extract the records you want into a file and then add from that file.

Examples
Notes

Examples

The first example shows a typical Add task. A self-describing file's records are added to the table called customer. This assumes that the self-describing file has the same structure and that the field names are the same as the column names.

```
>open oracle scott tiger {open SQL database}
>in custrecs             {input file you wish to add}
>add customer            {specify the Oracle table}
>exit                    {execute the task}
```

Our next example shows how to add by redefining the fields from a self-describing file into a table. The names are redefined so that the field names being extracted will match those in the table of the SQL database.

```
>open oracle scott tiger {open SQL database}
>in custrecs             {input file you wish to add}
>def cust_name,custname  {redefine the items to match}
>def cust_addr,address   {the names in the table}
>extract cust_name       {extract data under the column name}
>extract cust_addr
>add customer            {specify the Oracle table}
>exit                    {execute the task}
```

Our final example shows input from a flat file.

```
>open oracle scott tiger  {open SQL database}
>in salehist              {input file you wish to add}
>def cust_number,1,6,byte {redefine the items to match}
>def item_no,7,10,byte    {the names in the table}
>def sales,18,4,double
>extract cust_number      {extract data under the column name}
>extract item_no
>extract sales
>add customer             {specify the Oracle table}
>exit                     {execute the task}
```

Suprtool cannot currently support integers larger than two words.

Base Command   [BA]

The Base command is not available in Suprtool/UX.   See the Open command.

Before Command   [B]

Repeat any combination of the previous 1000 command lines, with or without editing.

```
        BEFORE      [ start [ / stop ] ]
                    [ string ]
                    [ ALL | @ ]
```

(Default: redo previous line)
(BQ=redo without change)

The Before command allows you to modify the commands before it executes them. If you don't need to change them, use BQ or Do.

The Before command uses Qedit-style Control characters for modifying the commands.   The default mode is to replace characters.   To delete use Control-D, and to insert use Control-B. If you prefer HP-style modify (D, R, I, and U), use the Redo command instead of Before.

Examples
Modify Operators
Persistence

Examples

```
>ll *.fd              {".sd" is not spelled right}
*.fd not found
>Before             {redo most recent command}
ll *.fd               {last command is printed}
     s                {you enter changes to it}
ll *.sd               {the edited command is shown}
                      {you press Return}

>listredo -10/
>before 5     {redo 5th command in stack}
>bef 8/10     {redo 8th through 10th}
>b ls         {redo last ls command}
>b ls *       {redo "ls *" command}
>b @*        {redo last containing "*"}
>before -2    {redo command before previous}
>before -5/-2  {redo by relative lines}
```

Modify Operators

If you wish to change any characters within the line, the modify operators are the regular Control Codes used in Qedit:

Any printing characters replace the ones above.
Control-D plus spaces deletes columns above.
(This assumes your EOF key has been altered from the HP-UX default)
Control-B puts you into "insert before" mode.
Control-A starts appending characters at the end of line.
Control-A, Control-D, plus spaces, deletes from the end.
Control-T ends Insert Mode, allowing movement to a new column.
Control-G recovers the original line.
Control-O specifies "overwrite" mode (needed for spaces).

Persistent Redo

Redo commands can be saved in a permanent file and can therefore be used from another session.   You can use the **Set redo** command to specify a filename to save your redo commands.   Please see the Set Redo command for details.

Chain Command [C]

The Chain command is not available in Suprtool/UX. See the Select command instead.

Count Option of Duplicate Command

Suprtool can count records that have duplicated sort keys.   See the Count option of the Duplicate command for details.

```
    >help duplicate
```

Define Command   [D]

Defines fields that can be used in the Duplicate, Extract, If, Item, Sort, Table, and Total commands.   With Define, you can do selection on ordinary data files using the same kind of readable "expression" that you use with databases.   You can also access data fields that are not actually structured as defined in the database (e.g., implicit subfields within an IMAGE field).

DEFINE   *field*, *definition*

field is an identifier up to 32 characters long, must begin with a letter, and can consist of letters A through Z, digits 0 through 9, or the following symbols:

```
 + - * / ? # % & @ _ $ '
```

In the case where the field name is written to a self-describing file, only the first 16 characters are used.

definition can be in two different forms: absolute or relative.

Absolute
Relative
Data-Types
Packed-Decimal
Data-Type Warning
Examples
Notes

Absolute Definitions

DEFINE *field*, *byteposition*,*sublen* [,*type*] [,*subcount*]

(Default: *type*=BYTE, *subcount*=1)

The *byteposition* is a positive integer giving the byte index where the field starts. The first byte is always number 1, not 0. The *sublen* is the number of bytes in the field. When the *subcount* is 1 (default), the *sublen* is the total number of bytes in the field. When you specify a *subcount*, the *sublen* is the byte-length of each subfield.

See **Data-Types** below for the definition of *type*.

<u>Examples</u>

```
>input uxfile,r 40, lf    {input from a disc file}
 >def qty,14,4,double       {double integer (PIC S9(9) COMP)}
 >def name,5,6              {character string of 6 bytes}
 >sort name                 {sort using the field "name"}
 >total qty                 {total all the values of the field}
 >exit                      {    "QTY"}
```

Relative Definitions

DEFINE *field, fieldname* [*(subscript)*]
[*[offset]*][*,sublen*][*,type*][*,subcount*]

(Default: *sublen*/*type*=same as *fieldname*)

The *fieldname* is a column from the table specified in Select, or a field from a self-describing file, or another Defined field. Relative definitions are similar to COBOL's Redefine verb.

The *sublen* and *type* are optional. They default to the total byte length and type of the *fieldname*. The (*subscript*) parameter is an optional sub-item index for arrays such as IMAGE compound items like 5J2 or 4X20. The first sub-item is number 1, and if no subscript is provided, Suprtool uses the first sub-item. The *[offset]* parameter is optional and specifies a byte offset from the position that would otherwise be used. This allows you to define fields relative to other fields. The *[offset]* starts at 1 and not at 0 (i.e., FIELD[1] is the first byte of the field).

To define a field that corresponds to the second index of the address array of the customer file, you would use:

```
>in customer             {self-describing file}
>define city,address(2)
>if city="Vancouver"
>list
>xeq
```

Data-Types

Here are the valid *types*:

BYTE	(printable ASCII characters)
INT/INTEGER	(two's complement)
DOUBLE	(two's complement)
IEEE	(IEEE floating-point)
PACKED	(packed-decimal)
PACKED*	(packed-decimal, last nibble unused)
DISPLAY	(zoned-decimal numeric field)
LOGICAL	(unsigned integer)
CHARACTER	(for Native Language Support)

The Define command also accepts Fpoint as the data-type for IEEE numbers.

<u>Image</u>

The following table shows the Suprtool definitions for the IMAGE data-types:

IMAGE Type	Bytes	COBOL Declaration	SUPRTOOL Definition
I1	2	S9(4) COMP	>define a,1,2,integer
I2	4	S9(9) COMP	>define a,1,4,double
I4	8	S9(18) COMP	>define a,1,8,integer
J1	2	S9(4) COMP	>define a,1,2,integer
J2	4	S9(9) COMP	>define a,1,4,double
J4	8	S9(18) COMP	>define a,1,8,integer
U*n*	*n*	A(*n*)	>define a,1,*n*,byte
X*n*	*n*	X(*n*)	>define a,1,*n*,byte
Z*n*	*n*	9(*n*)	>define a,1,*n*,display
P*n*	*n/2*	S9(*n-1*) COMP-3	>define a,1,*n/2*,packed
K1	2		>define a,1,2,logical
K2	4		>define a,1,4,logical
E2	4		>define a,1,4,ieee
E4	8		>define a,1,8,ieee

Data-type Display may have a trailing overpunch sign.

Packed-Decimal Fields

When defining packed-decimal fields, you must convert the number of decimal digits into a byte count. The last digit of a decimal field is *always* used for the sign. There are two data-types for decimal fields: PACKED for those that end on a byte boundary and PACKED* for those that end in the middle of a byte. Here are some example definitions of packed-decimal fields:

```
>define m,1,2,packed           {s9(3) COMP-3, P4 in IMAGE}
                                 {4 nibbles, last is sign digit}
>define n,1,2,packed*          {s9(2) COMP-3, P4 in IMAGE}
                                 {last digit is unused}
>define p,21,6,packed          {s9(11) COMP-3, P12 in IMAGE}
                                 {6 bytes, 3 words, 11 digits}
```

Data-Type Warning

The Define command accepts field definitions of any combination of byte-length and data-type. However, many combinations have limited usefulness in Suprtool. In these cases, Suprtool now prints a warning. For example:

```
     >def field,1,1,integer

Warning:  Length of 1 is of limited use for the data-type INTEGER
```

Examples

The following examples show the various data-types and combinations that are available with the Define command:

```
>define a,11,4,double              {J2 or I2, S9(9) COMP}
>define b,21,2,int                 {J1 or I1, S9(4) COMP}
>define e,21,8                     {character string}
>define f,address(5)               {fifth substring}
>define g,address(5)[3],4          {relative offset}
>define h,address(5)[4],2,byte     {middle of the address}
>define i,x[6],4                   {sixth byte of X}
>define j,invoice,6,byte           {redefine field as Byte}
```

Absolute
Relative

Absolute Example

The following example shows the most basic use of the Define command to create a new field at an absolute location in the input record.

```
     >def amt,11,2,int       {"amt" is an integer that starts}
     >output outfile         { at the 11th byte of file}
     >if amt > 1000          {"amt" is now a field}
```

Subcount

Absolute Example with Subcount

IMAGE and Suprtool allow fields to be repeated. In the next example, we define an amount field that repeats twelve times (e.g., once for each month of the year). We use a subscript when we want to refer to a specific month.

```
>def amt,11,2,int,12      {"amt" is an integer that repeats}
>output outfile           {    12 times}
>if amt(5) > 1000         {we select the 5th subfield which}
```

Relative Examples

Use the simplest form of relative definitions to rename existing fields.

```
    >def quantity,qty        {more readable name}
    >if quantity = "100 "    {selection on the new field}
```

The Define command copies the byteposition, sublength, and type to the new field, but it does not copy the subcount.   Define assumes that you want the first subfield:

```
    >def amount,amt         {amt is 12J2}
    >if amount > 1000       {amount is 1J2 and is the same}
    >out dinvoice           {    as amt(1)}
```

Subcount
Subscript
Offset

Relative Example with Subcount

Because Suprtool defaults the subcount to one, you might want to specify an explicit subcount when giving a new name to an existing field.

```
    >def amount,amt,,,12    {amt is 12J2; same length and type}
    >if amount(5) > 1000    {amount is 12J2; we are selecting}
    >out dinvoice           {    for May}
```

Relative Example with Subscript

Use subscripts to define a new field that corresponds to a specific subfield.

```
>def may,amt(5)     {amt is 12J2}
>if may > 1000      {may is the fifth subfield and}
>out dinvoice       {    it equals the May amount}
```

Relative Example with Offset

Many applications define subfields within a larger character field. A common example is a part-number where the first four digits are the warehouse location, the second four digits are the bin number, and the last four digits are a serially assigned number. Use the offset parameter to define new fields that are relative to the start of the part number. The file INVOICES is a self-describing file.

```
>in invoices                {part is 12 bytes}
>def warehouse,part,4       {warehouse starts at part}
>def bin,part[5],4          {bin is second four bytes}
>def release,part[9],4      {release is the last four bytes}
>if bin = "100"             {use any field for selection}
```

Note that redefining the digits of a larger numeric field only works when the field is a character field (type X, U or A).

Subscript

Relative Example Combining Subscripts and Offsets

If we have ten part numbers combined into one field (e.g., 10X12) we can still define a single field that corresponds to the bin number of one of the parts.

```
>in invoices                {allparts is 10X12}
>def bin3,allparts(3)[5],4
>if bin3 = "100"            {we are checking the fifth}
>out dinvoice               {   byte of the third part}
```

Notes

The purpose of the Define command is to tell Suprtool to look at a portion of the input record in a certain way. For example, if the record contains ASCII digits in the first ten bytes, Define could be used to assign a field definition to the first six bytes. In this case, the field must be defined as data-type Byte, Char or Display.

The Define command **cannot** be used to convert data from one format to another. Using the same example, the Define command could not be used to treat the first six digits of the ten digit ASCII field as Integer, Packed, Real, or any other numeric data format. If the input record contains ASCII digits, Define cannot change them to another data-type. (Note however that the Extract command may be used for data conversion. See the *Data Conversion* section of the Extract command for details.)

It may be helpful to think of the Suprtool Define command as similar to a COBOL REDEFINES clause, or a FORTRAN EQUIVALENCE statement.

Delete Command   [DEL]

The Delete command is not available in Suprtool/UX.

Do Command   [DO]

The Do command repeats (without changes) any of the previous 1000 commands.

```
DO    [ start [ / stop ] ]
      [ string ]
      [ ALL | @ ]
```

(Default: repeat the previous command)

Commands are numbered sequentially from 1 as entered and the last 1000 of them are retained.   Use the Listredo command to display the previous commands.   You can repeat a single command (`do 5`), a range of commands (`do 5/10`) or the most recent command whose name matches a string (`do list`).   If you want to modify the commands before executing them, use Redo or Before.

Examples
Notes

Examples

```
>listredo
>do                {do previous command again}
>do 39             {do command line 39 again}
>do 5/8            {do command lines 5 to 8 again}
>do list           {do most recent List command}
>do grep           {do last starting with "grep"}
>do grep job *     {do last "grep job *" command}
>do @job           {do last containing "job"}
>do -2             {do command before previous}
>do -7/-5          {do by relative line number}
>do 5/             {do command lines 5 to "last"}
```

Notes

The Do command can be abbreviated to , . , as in MPEX .   You cannot use ";" to combine commands on the same line.

Duplicate Command [DU]

By default, Suprtool copies all selected input records to the output file. The Duplicate command determines what to do with duplicate output records. Duplicate records can be discarded, producing an output file without duplicates. Alternatively, you may be interested in seeing the duplicate records, so you can create an output file consisting solely of the duplicate records. When deciding whether an output record is a duplicate, Suprtool either compares the keys only or the entire output record.

```
DUPLICATE   NONE | ONLY RECORD | KEYS [num] [ COUNT ] [ TOTAL ... ]
```

None

The None option removes duplicate records from the output file.   Suprtool compares each output record with the previous output record.   If they are not the same, the record is added to the output file.   This option corresponds to the former Nodup and Nodupkey options of the Output command.

```
>key 1,4
>duplicate none keys

 Input         Output

1111  10     1111  10
2222  25     2222  25
2222  35     3333  48
3333  48
```

Only

The Only option is the exact opposite of None. Only selects all output records that would not be written by the None option. When the Only option finds a record that duplicates a record already in the set, it writes that duplicate to the output file. Note that the first record is not written to the output file. Here are two examples:

```
>key 1,4
>duplicate only keys

 Input        Output          Input        Output

1111  10     2222  35        1111  10     2222  35
2222  25                     2222  25     2222  42
2222  35                     2222  35
3333  48                     2222  42
```

Record

Suprtool has two methods for comparing output records: Record and Keys.   The Record option compares the entire output record.   This option can be specified without a sort, but in that case the input file must already be sorted.   Note that there are two data fields in the records in the following example, so that a comparison of the entire record yields no duplicates.

```
>duplicate none record

 Input         Output

1111  10     1111  10
2222  25     2222  25
2222  35     2222  35
3333  48     3333  48
```

Keys

The Keys option only compares the sort keys to determine whether an output record is a duplicate.   This option requires that at least one sort key be specified.

```
>sort agent              {sort by agent}
>duplicate none keys
>output agents           {create roster of agents}
```

The Keys Num option determines the level at which Suprtool compares sort keys. This option controls which duplicate records get included in (or excluded from) the output file.

In the following example we sort by agent and by bill-date (in descending order), but only check for duplicates at the agent level.

```
>sort agent                {sort by agent}
>sort bill-date,desc       {sort by date }
>duplicate none keys 1     {only check for duplicate agents}
>output agents             {create roster of agents}
```

Count

The Count option causes Suprtool to produce a new field in the output record with the number of occurrences of each key value.   The count field is called st-count, and is an I2-type field.   The Count option can only be used with Duplicate None Keys.

```
>key 1,4
>duplicate none keys count

 Input       Output

1111  10    1111  10  1
2222  25    2222  25  2  {two records for key value 2222}
2222  35    3333  48  1
3333  48
```

Total

The Total option allows up to 15 fields to be subtotaled for each duplicate key. Separate the fields with spaces, not commas. The Total option can only be used with Duplicate None Keys. A new field is created at the end of the output record for each total. Each field is called st-total-*n*:

```
>sort customer-no
>extract customer-no
>duplicate none keys total sales-qty sales-amt
```

The above commands will create a self-describing file with the field customer-no and the total by each customer of the sales-qty in the field st-total-1. Similarly the field st-total-2 will contain the total sales-amt by customer number.

The following data-types are chosen for each total based on the data-type of the field:

Data-Type	Total Data-Type
IEEE	E4
all others	P28

Please see the P28 Fields section on how to define these fields in Cobol and PowerHouse programs.

Note that for byte fields, there can be only digits in the field. If there are other characters such as "+", "-", or ".", then Suprtool reports an error.
You can use the Link output option to easily see the fields that Suprtool creates. For repeated fields (e.g., 6I2), the first subfield is subtotaled if you don't provide a subscript.
You can combine the Count and Total options, but the Count option must appear before the Total option. You cannot combine the Total command with the Total option of the Duplicate command.

```
>key 1,4
>def field,5,2,integer
>duplicate none keys total field

 Input       Output

1111  10    1111  1  10
2222  25    2222  2  60  {25 + 35 = 60}
2222  35    3333  4  48
3333  48
```

Self-Describing Files

When you are using the Count or Total options, the sort information is not retained in the output file.   This means that if your output file is in LINK format, there are no sort keys.   If you are going to use this file from Suprlink, use the BY-clause of the LINK command to manually specify the sort keys.

Non-Self-Describing Files

If the input file is not a self-describing file and you are using Count or Total, you also need to extract all the fields from the file.   You can quickly extract the fields by defining a new field that contains the entire record. For example, if your records are 56-bytes long, then you do the following to extract all the fields:

```
>define entire,1,56,byte
>extract entire
```

Output Restrictions

When you are using the Count and Total options, the only Output formats that you can use are Data, Data-Num, Query, and Link.   If you want to quickly see the Count or Total results without using a second pass, use the List Standard command.

```
>select * from table
>sort key_field
>duplicate none keys count total data_field
>output result,link
>list standard
>xeq
```

Notes

The option of Only or None must be specified before the option of Record or Keys. Reversing the order causes a syntax error in the Duplicate command.

You cannot combine the Total command with the Total option of the Duplicate command.

Edit Command [ED]

The Edit command is currently not supported in Suprtool/UX.

Exit Command   [E]

Exit Suprtool in one of two ways.   Users are often frustrated when they exit Suprtool after specifying only part of a task because Suprtool starts processing the task.   To exit Suprtool without executing the current task, use the Abort option.

```
EXIT [ ABORT | XEQ ]
```

Typing Exit without parameters means Exit Xeq.   Suprtool recognizes special command names which specify both the Exit command and an Exit option (e.g., EA means Exit ABORT).

Abort
Xeq

Exit Abort [EA]

Cancels the current operation and terminates Suprtool.   The Exit command without parameters always attempts to perform the task currently specified, while Exit Abort cancels the task and terminates immediately.   Thus, Exit Abort is identical to Reset All;Exit.

Examples

Examples

```
>!#  You began to specify a sort, stopped for
>!#  coffee, and decided to cancel the task
>!#  upon your return.
>open oracle demo reader
>select * from customer
>sort name_last; sort name_first
... coffee break ...
>exit abort                {cancel the sort and terminate}
End Of Program
```

Exit Xeq [EX]

Signal the end of command input and the start of an extract operation. After the Suprtool task completes, Suprtool terminates. Exit Xeq is the default option (i.e., specifying Exit starts execution of the current task).

Examples
Notes

Examples

```
$/opt/robelle/bin/suprtool
>exit                        {no input was specified}
No action taken.

$/opt/robelle/bin/suprtool
>input rep23.newdata
>out rep23.data
>xeq                         {copy one file}
>input rep24.newdata
>out rep24.data
>exit                        {copy and stop}
```

Notes on Exit Xeq

If you have entered neither sort keys nor an input source, Exit terminates Suprtool without performing any task.   If you have defined an input source but without any sort keys, Suprtool does a copy operation prior to stopping.

Export Command   [EXP]

You cannot use Suprtool's Export command to invoke STExport/UX, but you can run STExport/UX by itself.

```
   /opt/robelle/bin/stexport

   STExport/UX/Copyright Robelle Solutions Technology Inc.  1988-2001
   (Version 4.4.13)
   $
```

STExport/UX only accepts self-describing files created by Suprtool/UX or the MPE SDUnix program.

Extract Command [EXT]

Assembles output records by stringing together fields extracted from input records. There can be up to 255 extracted fields, and the same field may be extracted more than once. Constant values may be used instead of the value of the field from the Input record.

EXTRACT *field* [(*subscript*)] [=*value* |=*field2*] [,...]

EXTRACT *field1* [(*subscript1*)] \ *field2* [(*subscript2*)]

EXTRACT *target-field* [=*expression*]

(Default: *subscript*=entire field)

Field Parameter

Each extracted *field* must be a database table column, or a field in an SD file, or a Defined field. If the field requires an Item definition, then the Item command must precede the Extract command. Extract specifies a rearrangement of the Input data fields to produce the Output data record; there is no data conversion, unless "output xxx, ascii" is specified.

The *subscript* parameter is valid only for compound items. The total item is extracted if it is compound and no *subscript* is specified.

```
>extract account         {extract the key value and}
>extract rating          {    one other field}
>output out1             {output file has two fields}
```

Cumulative Extracts

The Extract command is cumulative.   If two Extracts are specified in one run, all fields of the two Extract commands are used.

```
>extract status,balance,account,purchased
```

is equivalent to

```
>extract status,balance
>extract account,purchased
```

Constants

Extracting Constants

The *value* part of the Extract command is used to place a constant in each record of the output file.   In this case, the *field* defines the type and length that the *value* occupies.   The *value* portion must match the type of the *field*.   String values will be extended with blanks to fill the entire field. If the input data does not have a *field* of the correct size and type, you can create one using the Define command.

```
    >ext account        {key value}
    >ext rating=0       {place the constant value "0" in}
    >output out2        {    the "rating" field}
```

The total number of bytes that you can extract for all constants is 5,100 bytes for MPE/iX and HP-UX, and 1,275 bytes for MPE/V.

Packed and Display Constants

When extracting non-negative packed and display constants, Suprtool extracts them as unsigned unless you use a leading plus sign.   For the value zero, you can use a leading plus or minus sign, and you get a positive or negative 0.

```
>def field,1,6,packed
>ext field = 1     {unsigned 1}
>ext field = +1    {signed 1}
>ext field = +0    {positive 0}
>ext field = 0     {unsigned 0}
>ext field = -0    {negative 0}
```

Decimal Places

If a field has implied decimal places, Suprtool scales the input values according to the number of decimal places.   For example,

```
>item tax  ,decimal,2    {two implied decimal pts.}
>item total,decimal,2    {same}
>extract tax   = 1.02    {specified decimal pts.}
>extract total = 100     {value stored as 10000}
>output out3
```

Blank Fill

If you want to create an output record that consists of fifty blanks followed by the customer name, use:

```
     >define filler,1,50
     >ext filler=" "
     >ext name
```

String Constants

You can specify a string constant without referring to a field.   For example, to leave a space between fields, you must do the following:

```
     >extract account," ",rating
     >output *,ascii
```

Suprtool uses the length of the string to determine the size of the field. The following example extracts the same fields as the previous example, but each output record identifies the field:

```
     >extract "Customer Account=",account," "
     >extract "Credit Rating=",rating
     >output *,ascii
```

The output would look like:

```
     Customer Account=04598921 Credit Rating=      5000
     Customer Account=44657844 Credit Rating=     20000
     Customer Account=98753198 Credit Rating=      3000
```

The spaces after "Rating=", before the rating value, is due to the numeric field Rating being extracted with blanks for its leading zeros.   This is the result of the Ascii option of the Output command.

Repeated Fields

If the *field* is an IMAGE repeated field (e.g., 10J1), the Extract command places the value in each of the repeated fields when you do not specify the *subscript*. If you specify a *subscript*, only that one repeated field will have the new constant *value*.

In this example, Address is a repeated field (2X20). We wish to extract the data as it exists in the input record rather than forcing it to a constant value.

```
>ext account          {extract key value}
>ext address          {take both of the repeated}
>output out3          {    fields}
```

In the next example, we assume that the Balance field is a repeated field (12J2). We wish to make each of the repeated fields in the output record equal to 100:

```
>extract name
>extract balance=100
```

If we only wanted to extract the sixth field of BALANCE and set it to 100 we would do the following:

```
>extract name
>extract balance(6)=100
```

Character Constants

Use the ^-character to specify any ASCII character. The number (the actual ASCII value), or letter (^A means control A), must follow immediately after the ^-character. Suprtool treats character constants as strings. When you extract the constant to a field longer than one byte, Suprtool pads it with spaces.

```
   >define field,1,1     {byte field}
    >ext field = ^0       {binary zero}
    >ext field = ^G       {Control-G (bell)}
    >ext field = ^27      {escape}
    >ext field = ^252     {Roman-8 box}
    >ext field = ^186     {Euro currency symbol}
```

You can also extract the constant directly without referring to a defined field. This always produces a one-byte constant with no blank padding.

```
    >ext ^0           {binary zero}
    >ext ^13,^10      {Carriage Return, Line Feed}
    >ext ^M,^J        {CR, LF again}
    >ext ^27,"&dB"    {escape sequence}
```

Dates

Extracting Today's Date

To extract today's date, use the following:

```
     >item    field,date,ccyymmdd  {identify date format of field}
     >extract field=$today         {today's date}
     >extract field=$today(-1)     {yesterday's date}
     >extract field=$today(+1)     {tomorrow's date}
```

Use the Item command to qualify the field as a date. Suprtool uses the date format to determine the output format of the date. The $today function accepts one optional argument which is the number of days before or after today. The maximum number of days in either direction is 9999.

Oracle dates include both the date and the time. If you extract an Oracle date using $today, the time is always 00:00 (i.e., midnight).

Extracting Relative Dates

The Extract command provides the same relative date features as the If command (see the If command for a complete description of the options of $date). You must first use the Item command to identify the field name as a date. Suprtool uses the field type and length along with the date format to determine the output format of the date. Note that the three parts of $DATE are are always specified in (year/month/day) order, regardless of the date format of the field.

```
>item    field,date,mmddyy
>extract field=$date(*/*/*)       {today's date}
>extract field=$date(*/*-1/01)    {start of last month}
>extract field=$date(*/*-1/last)  {end of last month}
```

Oracle dates include both the date and the time. If you extract an Oracle date using $date, the time is always 00:00 (i.e., midnight).

$Stddate

Similar to the If command, the Extract command is also capable of utilizing the $stddate function.   This will allow for conversion of any of the supported Suprtool date formats to be converted to a date in the ccyymmdd date format in a double integer container.

For example,

```
     > get sales-detail
     > def new-ship-date,1,4,double
     > item ship-date,date,mmddyyyy
     > ext order-no / sales-amount
     > ext new-ship-date = $stddate(ship-date)
     > out salesinfo,link
     > xeq
```

Invalid Dates

Because the $stddate must have a valid date in order to properly convert the date to the ccyymmdd format, a value of 0 will be returned for any invalid dates.   An invalid date is any number of a particular date format whose date equivalent cannot be found on the calendar.

This means that if you attempt to extract use the $stddate function against a value that is not a valid date then the extracted value will be 0.

$Days

As with the $stddate function, the $days function is also available to the Extract command. You can convert any supported date to a Julian Day number in the following manner:

```
>in ordfile
>def ship-days,1,4,double
>def order-days,1,4,double
>def delay,1,4,double
>ext order-no
>ext ship-days=$days(ship-date)
>ext order-days=$days(order-date)
>ext delay=$days(ship-date)-$days(order-date)
>out neword,link
>xeq
IN=15, OUT=15.  CPU-Sec=1.  Wall-Sec=1.
```

Invalid Dates

If an invalid date is encountered, the extracted value will be zero. Therefore in the example above, if the order has not yet been shipped (ship-date does not contain a valid date) the resulting delay value will be negative.

Add and Subtract Dates

With the $days function, you can generate a date that is *n* days before or after any date. You only need to use two tasks: the first to generate the desired date in JulianDay format, and a second task to put that date into your format. In the following example, we show you how to get the previous day's date.

```
>input YOURFILE
>def origdate,1,8
>item origdate,date,yyyymmdd
>def jdate,1,4,int
>item jdate,date,julianday
>ext origdate
>ext jdate = $days(origdate) - 1   {or +7 for next week}
>out tmpfile,link
>xeq

>in tmpfile
>def yesterday,1,8,display         {your format here}
>item yesterday,date,yyyymmdd
>ext origdate
>ext yesterday = $stddate(jdate)
>out result,link
>xeq

Sample output:

ORIGDATE YESTERDAY

19990101 19981231
19991231 19991230
19990301 19990228
```

Date Limits

The $date function in Suprtool can generate dates between the years 1583 and 2583. Some date formats have limits based on their particular format, such as 2027 for a Calendar date and 2259 for the aammdd aamm, mmddaa, ddmmaa dates.

Range of Fields

Extracting a Range of Fields

You can specify a range of fields to extract using the following:

Extract Field1 \ Field4

This feature only works for self-describing files. If you specify a range, Suprtool extracts all 4 of the field names between field1 and field4 inclusive.

```
>in  sales          {a self-describing file}
>ext product_no\sales_qty
>out dsales
>xeq
```

is exactly the same as:

```
>in  sales          {a self-describing file}
>ext product_no     {first field in the range}
>ext product_price
>ext purch_date
>ext sales_qty      {last field in the range}
>out dsales
>xeq
```

Subscript
Alternate Syntax

Extracting a Range of Subscripted Fields

Suprtool accepts a subscript on either field in a range.   You can even use this feature to extract a range from a single field.   For example, if sales_amt is a 12J2 field:

```
     >in  sales             {a self-describing file}
     >ext sales_amt(4)\sales_amt(6)
     >out dsales
     >xeq
```

is equivalent to fc

```
     >in  sales               {a self-describing file}
     >ext sales_amt(4)        {first subscripted field}
     >ext sales_amt(5)        {intermediate subfield}
     >ext sales_amt(6)        {last subscripted field}
     >out dsales
     >xeq
```

Alternate Syntax for Extracting a Range

Suprtool accepts a slash "/" in place of the backslash "\" to specify a range. Use the slash with care, because it is a valid character in field names.   For example,

```
     >extract a/b
```

would produce the error message:

```
     Error:  Field "A/B" does not exist
```

To use a slash in an extract range, surround it with spaces:

```
     >extract a / b
```

Numeric Expressions

You can specify arithmetic expressions for any numeric data-type in the Extract command. Arithmetic expressions involve the operators +, -, *, / and mod. Extract arithmetic expressions work exactly as If command arithmetic expressions. To extract an expression, use this syntax,

EXTRACT *target-field* = *expression*

Target-Field

The *target-field* determines the byte-length, data-type, and repeat-count for the expression. The expression is extracted during the output phase and cannot be used by other Suprtool commands that accept fields (e.g., sort).   To avoid confusion, it is best to define a new field name for the *target-field* instead of using an existing field name.

Examples

```
>extract budget99 = actual98 + 1000

>extract total = cost * qty

>extract day = ccyymmdd-date mod 100
```

In the following example, the field `total` is used twice. In the first case, it is used to tell Suprtool how to format the arithmetic expression. In the second case, it is used in the sort command. **Warning:** In this example, the output file is sorted by the value of `total` as it appears in the input record. It is not be sorted by `cost * qty`.

```
>extract total = cost * qty
>sort    total          {sort by input total}
```

Restrictions

You can only use one expression in each Extract command, and the expression must be the last item.   If you want to extract several expressions or more fields after an expression, you need to use several Extract commands.

Incorrect

```
>extract name, i=sales + tips, c=cost + expense, dept
```

Correct

```
>extract name, i=sales + commission
>extract c=cost + expense
>extract dept
```

Constants vs. Expressions

If you have an arithmetic expression that starts with a constant, Suprtool assumes that you are attempting to extract a single constant value and not an arithmetic expression. To specify an arithmetic expression that starts with a constant, surround the expression with parentheses. For example,

Incorrect

```
>extract c = 6000 - cost

Error:  Missing comma or invalid arithmetic expression
```

Correct

```
>extract c = (6000 - cost)
```

Numeric Truncation

The accuracy of arithmetic computations is limited to approximately sixteen digits. Suprtool may truncate four-word integers (quad), or large packed-decimal numbers, or display numbers when they are converted to floating-point.   Suprtool does not produce any error or warning in this case.

Division by Zero

Suprtool reports an error in the input record number if an arithmetic computation results in division by zero. Use Set Ignore On to force Suprtool to ignore division by zero errors. With Set Ignore On, the result of division by zero is zero.

The speed of a task decreases when you ask Suprtool to ignore many division by zero errors. It is better to check for zero in the If expression before using it in division.

```
>if qty <> 0      {avoid division by zero}
>extract average = total / qty
```

$Abs function

Suprtool supports an $abs function, which returns the absolute value of a number.   For example, if a field called Credit contains the value -547.83, the $abs function returns 547.83.

This function will work on a field or even on an expression such as:

```
>def newcredit,1,4,double
>ext newcredit = $abs(credit / 100 * 1.07)
```

This function will also work in the If command:

```
>if $abs(credit / 100 * 1.07) > 500.00
```

$Truncate function

Suprtool supports a $truncate function which returns the number to the left of a decimal place.   For example if the field stddev contains the value 547.83, the $truncate function will return 547.   Note that there is no rounding.

This function will work on fields and expressions.   For example,

```
    >def newdev,1,4,double
    >ext newdev = $truncate(stddev / 100 * 1.07)
```

This function will also work in the If command:

```
    >if $truncate(stddev / 100 * 1.07) > 200
```

String Expressions

You can combine byte-type fields together and use the built-in string functions to create new fields out of existing ones.   This can reduce the number of tasks required to provide a solution.   String expressions may involve the + operator and $upper, $lower $trim, $ltrim or $rtrim.   To extract a string expression, use this syntax:

EXTRACT *target-field = expression*

Target-Field

The *target-field* determines the byte-length for the expression.   The data-type must be Byte or Char.   The expression is extracted during the output phase and cannot be used by other Suprtool commands that accept fields (e.g., Sort).

Examples

Examples

```
>extract id-no = warehouse-no + bin-no

>extract full-name = first-name + last-name
```

Constants vs. Expressions

If you have an string expression that starts with a string, Suprtool assumes that you are attempting to extract a single string value and not an string expression. To specify a string expression that starts with a constant, surround the expression with parentheses. For example,

Incorrect

```
>extract name = " " + product-desc

Error:  Missing comma or invalid arithmetic expression
```

Correct

```
>extract name = (" " + product-desc)
```

Variable Length Strings

String expressions use variable-length strings. Suprtool keeps track of the length of every string, and all operations are done using the actual string length. For fields, the length of the string is the length of the field. If you do not want to retain all the spaces in a field, use one of the built-in trimming functions.

String constants are created with the exact length of the constant. For example, the string "abc" is three characters long and the string "a" is one.

When assigning the string expression to the target field, Suprtool pads the final string value with spaces to fill out the target field. String expressions longer than the target field generate an error.

```
>in testfile
>def a,1,10,byte
>ext a="I'm too long for this container"

Error:  String is too long for the specified item
```

String Truncation

Suprtool produces an error if the string expression is longer than the target field. You cannot override this error with Set Ignore On. To help avoid the error, you may want to trim the expression of trailing spaces before assigning it to the target field. For example,

```
>extract new-field = $trim(a + b + c)
```

Upshifting Strings ($Upper)

Use the built-in function $upper to upshift all the characters of a string expression into uppercase characters. This function can be used to upshift a single field, a complicated string expression, or any subpart of an expression. Both ASCII and Roman-8 characters are upshifted by $upper. For example,

```
>extract city-up = $upper(city)

>extract full-name = $upper(first + last)

>extract desc = desc-1 + $upper(desc-2)
```

Downshifting Strings ($Lower)

If you want to downshift all characters of a string expression to lowercase, use the built-in function $lower.   This function can be used to downshift a single field, a complicated string expression, or any subpart of an expression.   Both ASCII and Roman-8 characters are downshifted by $lower.   For example,

```
>extract city-lower-case = $lower(city)

>extract city-state = $lower(city + state)

>extract desc = desc-1 + $lower(desc-2)
```

Trimming Spaces ($Trim, $LTrim, $RTrim)

Use one of three built-in string functions to remove leading or trailing spaces from a string expression. The three functions are:

$Trim:	Remove leading and trailing spaces from the string expression.
$LTrim:	Remove leading spaces.
$RTrim:	Remove trailing spaces.

Data Conversion

You can convert numeric fields from one data-type to another. Any nonbyte field type is considered to be numeric. You can also lengthen or truncate character fields. The general syntax for doing conversions is:

EXTRACT *target-field = source-field*

Target-Field

The *target-field* determines the byte-length, data-type, and repeat-count for the expression. The expression is extracted during the output phase and cannot be used by other Suprtool commands that accept fields (e.g., Sort).   To avoid confusion, it is best to define a new field name for the *target-field* instead of using an existing field name.

The following example shows defining a new *target-field* as a double integer. The Extract command *target-field* then takes the definition from the Define command and extracts data from the *source-field* (display-field).

```
>in oldfile
>def salesqty,1,4,double
>ext order-no / order-date
>ext salesqty = display-field
```

Packed and Display Fields

When the target of an extract conversion is a packed- or display-type field, Suprtool always converts positive values to a neutral packed- or display-value.   To ensure that expressions with positive values have a positive result, use the $signed function:

```
     >extract packed_field = $signed(int_field)

     >extract display_field = $signed(dbl_field / 10)
```

Truncation errors can occur when Suprtool converts from nonfloating-point to floating-point. See the discussion under "Arithmetic Expressions" above.

Byte Fields

Use the Extract command to shorten or lengthen byte-type fields.

If the *target-field* is longer than the *source-field*, Suprtool fills the trailing space in the *target-field* with spaces.

Byte to Numeric Conversion

Suprtool cannot explicitly convert from a byte field to a numeric field such as a double integer. The Extract command, however, does allow conversion from a display field to a double integer (or any other numeric field).

You can define a byte field to be a display field if all of the characters in the field contain a number. For example if you have a six-character byte field that looks like this:

```
     012345
```

you can define it in the following manner:

```
     >def display-field,1,6,display
```

This field can then be converted to any of the other numeric types that Suprtool supports.

If the field is six characters and contains blanks, you would have to pad the field with zeros so that Suprtool can manipulate the field as a display field.

Restrictions

You can only use one expression in each Extract command, and the expression must be the last item. If you want to extract several expressions or more fields after an expression, you need to use several Extract commands.

Incorrect
```
>extract name, i=sales + tips, c=cost, dept
```

Correct
```
>extract name, i=sales + commission
>extract c=cost
>extract dept
```

Extracting Bits

The Extract command can be used to define individual bits from one data item as separate fields.

```
>def order-shipped,1,2,int
>def order-paid   ,1,2,int
>ext order-shipped=status-field.(0:1)
>ext order-paid=status-field.(1:1)
```

This makes it easier to check the status of certain bits within a given field.

EBCDIC Conversions

Use the $etoa or $atoe functions to convert specific fields from EBCDIC to ASCII or vice versa. Each of these functions accepts a single parameter that is a byte-type field:

```
Extract $ETOA(char-field)
Extract $ATOE(char-field)
```

There are several restrictions on the $etoa and $atoe functions:

¤ They do not work with either the ASCII or PRN output options.

¤ You cannot extract an EBCDIC constant. The following example would produce an error message:

```
>extract $etoa(char-field) = 'abcdef'
```

¤ You cannot extract a range of fields using $etoa or $atoe.

Notes

The Extract command is valid only with

```
Output xxx,data
Output xxx,data,num
Output xxx,query
Output xxx,link
Output xxx,ascii
Output xxx,prn
```

The Extract occurs logically after the sort phase, if any, but prior to the final Output, Put, or List. An If command can refer to fields of the input record that are not included in the extracted output record. The sort keys can be fields that are not among those extracted.

If the extracted record length is shorter than the input record length, Suprtool attempts to speed up sorts by doing the extract before sorting. Suprtool can only do the extract before sorting if the output option is DATA (the default), QUERY, or LINK, and all of the sort keys are included in the Extracted fields.

Form Command   [F]

The Form command displays information about an SQL database, or the current Select command, or the fields in a self-describing file.

```
FORM    [ filename ]
```

(Default: depends)

When showing the form of an SQL select, Suprtool shows the column's name, the SQL type and the Suprtool type.   When showing the form of a self-describing file, Suprtool shows the byte offset of each field after the subcount, type, and sublength.   The first field always appears at offset one.

Example

```
>open oracle scott tiger
>select * from emp
>form
 Column Name:   Oracle Type:      Nulls:     Suprtool Type:

 EMPNO           Number   (4)        N         Integer
 ENAME           Varchar2 (10)       Y         Byte
 JOB             Varchar2 (9)        Y         Byte
 MGR             Number   (4)        Y         Integer
 HIREDATE        Date                Y         Oracle Date
 SAL             Number   (7,2)      Y         Packed
 COMM            Number   (7,2)      Y         Packed
 DEPTNO          Number   (2)        Y         Integer
```

SQL Database

If an Allbase database is open and no input file has been specified, the default Form command shows all of the tables in the database. If a Select command has been specified, the default Form command shows the columns in the Select command. The exact format of the Form command is different for each SQL database.

Self-Describing Files

The Link output option produces an SD file with information about how the file was sorted, what fields are compound, and the date format or the number of implied decimal places for any fields.   The Form command shows all of this information:

```
 >form custfile

File: custfile   (SD Version B.00.00)  Has line feeds
   Entry:                     Offset
      CHAR_FIELD           X5      1  <<Sort #1>>
      REPEATED_I1         3I1      6  {compound field}
      DATE_FIELD           J2     12  <<YYYYMMDD>>
      COST_FIELD           J2     16  << .2 >>
Entry Length: 20  Blocking: 1
```

Default Form

If a Select or Input command of a self-describing file has been entered, a Form command without parameters shows the fields in the current input source. If an Open command has been specified, but no input source, a Form command without parameters shows the tables in the SQL database.

Get Command   [G]

The Get command is not available in Suprtool/UX.   See the Select command instead.

Help Command [H]

Show what commands and options are available in Suprtool.

```
HELP   [ command | keyword [ ,option ] ]
```

(Default: browse through the entire help file)

Commands
Keywords
Quick
Notes

Command Help

If you specify any parameters, Help first assumes that you want help on a Suprtool command. If you know the structure of the help file, you can additionally specify one of the keywords under the command name.

```
>help ext                {help on the Extract command}
>help ext,notes          {Notes section of the Extract command}
```

Keyword Help

If we cannot find any help in the "Commands" section of the help file, we assume that you specified one of the outer-level keywords in the help file. To see this list of keywords, type help with no parameters.   You see a short introduction to Suprtool and then a list of keywords.   You can specify any of these keywords on the Help command.   You can also specify a subkeyword.

```
>help start             {Quick Start section}
>help start,task        {Task section of Quick Start}
```

Quick Help - HQ

HQ asks Suprtool to look under the keyword Quick in the help file.   Quick contains the text from the Suprtool Quick Reference Guide, offering the experienced user a quick review of the syntax of any command.

```
>hq input                    {quick description of Input}
>hq commands                 {quick list of command names}
```

Notes

If no parameters are specified, Help allows you to browse through the "help" file.   The Help Command uses the QHELP subsystem to allow you to look at the material in the file /opt/robelle/help/suprtool (which contains most of the user manual).   For "help in help", type "?" when you see the QHELP prompt character ("?").   The help file is organized into levels. To go back to the previous level, press Return instead of a key name.   If you press F8, you will exit the QHELP subsystem and return to Suprtool.

If Command [IF]

Specifies a subset of records to select from the input source during the next extract task. The If command supports full logical expressions, with comparisons between all data-types, between data fields and constant values, or between one data field and another. The If command also provides partial string compares, bit field extracts, subscripted IMAGE fields, AND-OR-NOT operators, and parentheses to override precedence. You can use arithmetic expressions involving any numeric data-types.

IF *expression*

Note: The examples below show multiple If commands. These are for illustrative purposes only. Suprtool does not permit multiple If commands in a single task. Instead you can combine multiple conditions using AND and OR.

Alternatives to the If command

There are a few selection criteria that the If command cannot perform. In these cases, you need to use other Suprtool commands. If you wish to select by record numbers, use the record number options in the Input command. If you wish to limit the number of records selected, use the Numrecs command.

There Is No Else Clause

The If command in Suprtool does not have an Else clause. To select the records that do not match the If criteria, use a second task with the same criteria negated by a NOT.

```
>get    input                {this task is the "If ...  then"}
>if     expression
>output file1
>xeq

>get    input                {this task is the "else"}
>if     not (expression)
>output file2
>xeq
```

Expression
Constants
Subscripts
Numeric Expressions
String Expressions
Dates
Long Expressions

Expressions

An *expression* specifies the logical criteria that Suprtool uses to select records from the input source.

Simple Expressions

The simplest *expression* is a single *comparison* between two fields (e.g., A=B) or a field and a constant (e.g., A="XX").

```
field        relation  field
field        relation   constant
```

Fields

A field can be a temporary, Defined field, or a field from a self-describing file, or a column from a database table. Each *field* has a type (see the Key command for further details). The *constant* must match the type of the *field*. If the *field* has a byte-type, you must surround the *constant* with quotes.

```
    >if name="MARIE REIMER"   {name is byte}
     >if rating>10000          {rating is integer}
     >if balance=arrears       {compare two fields}
```

Constants

A *constant* is a value that matches the data-type of *field*. Constants are either a string constant in quotes, a numeric constant, or a date constant specified with $date or $today. See the next section about *Constants* for more details.

`"KERRY LATHWELL"`	{string constant}
`12345`	{numeric constant}
`$date(00/07/09)`	{date July 9, 2000}

Relations

A *relation* is one of the size comparison symbols (Suprtool does not use words like "EQUALS" as in QUERY):

=	equal to
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
<>	not equal to

Complex Expressions

Complex *expressions* can be made by combining the AND, OR, and NOT operators, arithmetic operators (+, -, *, / and mod), and parentheses.   The order of precedence of operators, from highest to lowest, is

(...)	Highest.
NOT	Take the opposite.
AND	Both must be true.
OR	One or the other must be true.
-	Unary minus.
* /	Higher than addition and subtraction.
+ -	Use parentheses where necessary.

```
>if status="1" and amount>100 or purchased="000115"
>if (status="1" or status="2") and amount>100
```

Multiple Values

You can check a data field for several test values without using the AND and OR operators. List the alternate values separated by commas. The OR operator is = (equal sign). Instead of "IF A=5 OR A=6 OR A=7", use "IF A=5,6,7". This selects a record if A is equal to 5, 6, or 7. The AND operator is <>. Instead of "IF A<>5 AND A<>6 AND A<>7", use "IF A<>5,6,7". This selects a record if A is anything but 5, 6, or 7.

```
>if field = 5,6,7
>if part = "12345","67890","39201","92308","14892"
>if delivered <> 981231,990101
```

This method works if you are searching for a small number of values. Use the $lookup function to check a data field for many test values. You can also apply the NOT operator to find data that are not in a table.

IF $LOOKUP(*tablename*, *fieldname*)

You can also apply the NOT operator to find data that are not in a table.

IF not $LOOKUP(*tablename*, *fieldname*)

The $lookup function returns TRUE, if the specified field name contains a value from the specified table. You can also look for values that are not in a table.

IF NOT $LOOKUP(*tablename*,*fieldname*)

See the Table command for a complete description of how to combine tables and the $lookup function.

tablename

The name of a table specified in the Table command.

fieldname

A field from the input record. This field must be exactly the same length as the *item* used in the Table command.

<u>Performance of $lookup</u>

Performance of $Lookup

The $lookup function can be quite slow when you are searching huge tables. Because the If command uses short-circuit evaluation, $lookup should be specified as the last part of the If command.   For example,

```
     >if status = "10" and $lookup(cust-table,account)
```

is faster than

```
     >if $lookup(cust-table,account) and status = "10"
```

$Null(fieldname)

The If $null(*fieldname*) command selects any rows that have null values in them.   This feature is available only for SQL databases and only on columns that allow null values:

```
    if $null(SALESTOTAL)
```

If you want to find only those values that are not null, you can add the NOT keyword in front of $null:

```
    if not $null(SALESTOTAL)
```

Constants

This section describes numeric and string constants.   For details on using date constants, see the section *IF (Selection by Date)*.

Numeric Constants

Numeric constants are not enclosed in quotes. Numeric constants may be just simple numbers (e.g., 5 0 -56 10004) or they may have a decimal point (e.g., 5. 0.0 -.56 99.9 1.4). IEEE numbers may also have a scale factor (e.g., 5E-5 0.01E+4). "Over-punches" for the sign are not required, or recognized, in Suprtool. Always enter -11 as -11, not 1J for a DISPLAY field.

String Constants

String constants are delimited with double- or single-quote marks.   That is, either "VANC" or 'VANC'.   Any characters within quotes are not upshifted.   If the constant is shorter than the field to which it is being compared, the constant is padded with blanks.   String constants are expected for fields of type BYTE, U, or X, but numeric constants are expected for fields of type Z (zoned decimal).

```
     >if field = " "          {check for all blanks}
```

If you want to compare for a quote itself, you include two quotes in the string for each quote you want.   For example, "XX""XX" means to look for XX"XX.

```
     >if field = "XX"         {double-quotes are okay}
     >if field = 'XX'         {so are single-quotes}
```

Character Constants

Use the ^-character to specify any ASCII character. The number (the actual ASCII value), or letter (^A means control A), must follow immediately after the ^-character. Suprtool treats character constants as strings. When you compare the constant to a field longer than one byte, Suprtool pads the constant with spaces.

```
>define field,1,1      {byte field}
>if field = ^0         {binary zero}
>if field = ^G         {Control-G (bell)}
>if field = ^27        {escape}
>if field = ^252       {Roman-8 box}
```

To look for "null values" or "low values" in byte-fields, it is usually sufficient to check the first byte:

```
>define byte1,bigfield,1,byte
>if byte1 = ^0
```

Subscripts

Use subscripts to access individual items in repeated fields, or to access substrings.

Numeric Subscripts

For repeated numeric fields only one index is allowed.   If Table has the form 10J2, it holds ten double integers.   Table(1) is the first sub-item, Table is the same as Table(1).   Table(5) is the fifth sub-item, ...

```
    >if table(5) = 23
    >if table(2) = 20 or table(4) = 30
    >if table(8) = 31 and table(9) = 28
```

Character Subscripts

Character string fields may have 1, 2, or 3 subscripts after them.   Character string fields are allowed more than one subscript value.   If ADDR has the form 5X30, it consists of 5 substrings of 30 characters each.   ADDR(1) is the first 30-character sub-item of ADDR. ADDR without subscript is the same as ADDR(1). ADDR(2,4) is the second sub-item, starting with the 4th byte and extending for the remainder of the sub-item, 27 bytes.   ADDR(2,4,6) starts at the same location, but extends for only 6 bytes.

If NAME has the form X50, it is not a repeated field.   In this case, NAME is the same as NAME(1).   NAME(1,4,6) is the first (and only) sub-item, starting at the 4th byte and extending for 6 bytes.   NAME(1,10) is a field that starts at the 10th byte and implicitly extends for 41 bytes to the end of the field.

```
>if name(1,4,6) = "HAWAII"
>if addr(3)="VANCOUVER, B.C."
>if addr(3,11,20)=="@B.C.@"   {pattern matching}
```

Numeric Expressions

Bit Extracts

The If command can extract and test any series of one or more contiguous bits in a field. Suprtool allows bit extracts only on Integer or Logical fields of two bytes in length (one word). To do a bit extract from another type of field, first use Define to redefine the data as a two-byte Logical field.

Once Suprtool extracts a bit string, it always treats it as an Unsigned Integer, a Logical, and never interprets it as negative. The format for bit extracts calls for a starting bit number and a bit count. The 16 bits in a computer word are numbered from the left, 0 to 15. The two bytes to extract from need not be on a "word boundary" (i.e., they can start in any byte position). See the Define command for how to define a two-byte logical field.

field . (*startbit* : *bitcount*)

```
>define bitfield,name,2,logical
>if bitfield.(4:2)=3
```

Check-Byte

How to Check a Byte for a Numeric Value

Since Suprtool does not have one-byte integers, it can be difficult to check a single byte for a specific numeric value.   Use a two-byte integer Define field and the bit-extract operator to solve this problem:

```
    >define word,transcode,2,integer
    >if word.(0:8)=13
```

See *Character Constants* for an alternate method.

Decimal Places

Use the Item command to specify the number of implied decimal places in an item. If you do not do this, you must scale all numbers in the If command. For example, let's assume that you want to find all inventory records with a cost equal to $80.59. If you do not use the Item command, your If command would look like this:

```
     >if cost = 8059              {no decimal places}
```

By telling Suprtool about the number of decimal places in the cost item, your If command looks more natural (which usually means you will make fewer mistakes):

```
     >item cost,decimal,2
     >if cost = 80.59             {decimal places included}
```

Numeric Conversion

The If command can compare two different numeric fields (not just one field to a constant). All relation operators are supported: <, <=, =, <>, >, and >=. However, you cannot compare a byte-field to a numeric-type field.

Suprtool usually converts the field on the left side of a relational operator to floating-point. Then the floating-point number is converted into the type of the field on the right side of a relational operator and the comparison is done. The exceptions to this rule are integer-to-double, packed-to-packed, and display-to-display comparisons, which use a direct comparison algorithm.

Truncation errors can occur when Suprtool converts from one field type to floating-point. See the discussions under *Accuracy* and *Numeric Truncation*.

Arithmetic Expressions

You can specify arithmetic expressions for any numeric data-type in the If command. Arithmetic expressions involve the operators +, -, *, / and mod. The Mod operator returns the remainder between a dividend and a divisor. Arithmetic expressions cannot start with a numeric constant (e.g., if 2 + a = 10 is invalid).   Arithmetic is not allowed on byte-type fields.   If you have a byte-type field that consists entirely of numeric digits, redefine the field as display type and use the redefined field name in the If command.

Example
Division By Zero
Missing Features

Examples

```
>if field + 10 = 1115      {numeric field}
>if cost * qty > 10000
>if total < qty * price + tax
>if yymmdd-date / 100 mod 100 <= 03 {first quarter}
```

Division by Zero

Suprtool reports an error and the input record number if an arithmetic computation results in division by zero.   Use Set Ignore On to force Suprtool to ignore division by zero errors.   With Set Ignore On, the result of division by zero is zero.

Your task executes more slowly if you have a lot of division by zero errors and you have asked Suprtool to ignore them.   A better approach is to check for zero in the If expression before using it in division:

```
>if $read
-   qty <> 0  and       {avoid division by zero!}
-   total / qty > 100
-   //
```

Missing Features

Arithmetic overflow in computations will cause Suprtool to abort.

Accuracy

By default, Suprtool uses floating-point arithmetic to compute.   In some cases, there can be slight inaccuracies due to rounding errors.

Numeric Truncation

The accuracy of arithmetic computations is limited to approximately sixteen digits. Suprtool may truncate four-word integers (quad) or large packed-decimal or display numbers when they are converted to floating-point. Suprtool does not produce any error or warning in this case.

$Abs function

Suprtool supports an $abs function, which returns the absolute value of a number. For example, if a field called Credit contains the value -547.83, the $abs function returns 547.83.

This function will work on a field or even on an expression such as:

```
     >if $abs(credit / 100 * 1.07) > 500.00
```

This function will also work in the Extract command:

```
     >def newcredit,1,4,double
     >ext newcredit = $abs(credit / 100 * 1.07)
```

$Truncate function

Suprtool supports a $truncate function which returns the number to the left of a decimal place.   For example if the field stddev contains the value 547.83, the $truncate function will return 547.   Note that there is no rounding.

This function will work fields and expressions:

```
     >if $truncate(stddev / 100 * 1.07) > 200
```

This function will also work in the Extract command:

```
     >def newdev,1,4,double
     >ext newdev = $truncate(stddev / 100 * 1.07)
```

String Expressions

You can do comparisons with byte-type fields in numerous ways using Suprtool. These powerful features minimize the number of tasks you must execute in order to select the data you need.   The fewer the number of tasks, the faster your data is delivered to the users and applications that need it.

You can combine byte-type fields together and use the built-in string functions to create string expressions.   String expressions involve the + operator and the other string functions such as $lower, $upper, $trim, $ltrim and $rtrim.

Fixed vs. Variable Length Strings

String comparisons are done using fixed- and variable-length strings. For most users, there should be no difference between the two types of strings. When doing string comparisons, Suprtool always pads shorter strings with spaces, with the one exception of comparing two fixed-length fields (see "Byte Fields" below).

String expressions involving the + operator or the $upper, $lower $trim, $ltrim and $rtrim built-in functions are done using variable-length strings. Suprtool keeps track of the length of every string, and all operations are done using the actual string length. For fields, the length of the string is the length of the field. If you do not want to retain all of the spaces in a field, use one of the built-in trimming functions.

When creating string expressions, string constants are created with the exact length of the constant. For example, the string "abc" is three characters long and the string "a" is one.

Byte Fields

For historical reasons, comparing two byte-type fields to each other is a special case. If the two fields are exactly the same length, Suprtool compares them completely. If one field is shorter, the comparison is done for the length of the shortest field. Suprtool does not check for spaces in the trailing characters of the longer field. For example,

```
>define  short, 1,10   {ten character field}
>define  long ,11,15   {fifteen character field}
>if      short = long
```

In this example, Suprtool compares the ten bytes in the short field with the first ten bytes of the long field, but ignores the last five bytes of the long field. If the expression on either side of the equal sign consisted of more than one field (using the + operator) or involved any of the string functions, ($upper, $lower, $trim, $ltrim or $rtrim), Suprtool would have compared both sides of the equal sign by padding the shorter field with spaces. It is only the case where you are directly comparing one byte-type field to another that Suprtool uses the length of the shortest field for the comparison.

You cannot compare a byte-field to a numeric-type field. If you have a byte-field that consists entirely of numeric digits, redefine the field as a display-type and use the redefined field name in the If command.

Character Type

Byte-type fields can also be checked to see whether they contain only Alpha, Numeric, Alphanumeric, or Special characters. The complete field is compared against the specified character types.

Alpha	A-Z, a-z (52 characters)
Numeric	0-9 (10 characters)
Special	anything else (194 characters, including spaces, punctuation, Roman-8 letters, binary junk)

For the test result to be true, **all** the characters in the field must be of the specified character type. To test a substring, use the Define command to define a subfield.

```
>if field = alpha
>if field <> numeric
```

Examples:

`"1234"`	{numeric}
`"12.3"`	{no class, contains both numeric and special}
`"ABCD"`	{alpha}
`"B JONES"`	{no class, contains both alpha and special}
`"    "`	{special}
`"A1B2"`	{alphanumeric}

Pattern Matching

Suprtool can also select records based on a pattern of characters, rather than an exact string of characters. For example, use the following to select all records with "ROBERT" anywhere in the Name field,

```
     >if name == "@ROBERT@"
```

The double equals (==) is the operator for pattern matching. The at sign (@) means anything before or after "ROBERT" is acceptable, including nothing.

For character fields, there are two comparison operators for patterns: "==" (matches), and "><" (does not match). The pattern is specified as a quoted string, using the special characters listed below. Embedded spaces are allowed in the pattern and must be matched in the target field.

Special Char

These are the special characters:

@	Zero, or more, characters of any type.
#	A single numeric character.
?	A single alphabetic or numeric character.
~	Zero, or more, blank characters.
&	Escape character (&@ looks for the @ character).
^	Reserved for future use.
!	Reserved for future use.

Any other character must be matched, one for one.

```
>if name=="@BIRD@"       {does name contain BIRD anywhere?}
>if name=='@JIM@BIRD@'  {does name contain JIM, perhaps}
                         {other characters, then BIRD?}
>if name><"@#@"          {does name not contain numerics?}
>if name=='@qedit@','@suprtool@'  {qedit or suprtool?}
```

For more information, see "Special Characters" in the Glossary.

Trimming Spaces ($Trim, $Ltrim, $Rtrim)

Use one of three built-in string functions to remove leading or trailing spaces from a string expression. The three functions are:

$Trim:	Remove leading and trailing spaces from the string expression.
$LTrim:	Remove leading spaces.
$RTrim:	Remove trailing spaces.

Because Suprtool pads shorter strings with spaces when doing comparisons, trimming spaces is most useful when creating a combined string with several fields. For example, you might want to combine a person's first and last name (including a space between the two):

```
     >if $trim(first) + " " + $trim(last) = "Joe Smith"
```

Mixed Case ($Upper and $Lower)

By default, Suprtool does an exact match when comparing two string expressions. If the expressions vary in the capitalization of characters, Suprtool finds them to be different. To do caseless string comparisons or pattern matches, use the $upper or $lower functions. Both ASCII and Roman-8 characters are shifted by $upper and $lower. For example,

```
>if $upper(city) = "VANCOUVER"

>if $lower(city) = "edmonton"
```

Note that if you use the $upper or $lower functions, Suprtool does not upshift or downshift any constants used in the comparison. You must explicitly specify the constants in the correct case or you can use $upper or $lower with the constant:

```
>if $upper(city) = $upper("vancouver")
```

Use the $upper or $lower functions for caseless pattern matching. As with other comparison operators, you must specify constants in the correct case when doing pattern matching:

```
>if $upper(city) == "VAN@"

>if $lower(city) == "ed@"
```

You can use $upper and $lower with string expressions that combine many fields and string functions as shown in the following example:

```
>if $read
-   $upper($trim(first) +
-             " "             +
-             $trim(last))
-   = "JOE SMITH"
-
```

Date Selection

The If command has four functions to help select records based on dates: $date, $today, $days and $stddate.   The $date function works for any date. The $today function works for the current date and dates relative to today. The $stddate and $days functions work for almost any date.   To use these date functions, you must first identify the date format of an item by using the Item command.

The $date function makes it easier to specify a target date for certain date formats (e.g., PHdate or ASK). To select records based on a specific date, use this feature:

```
>if field=$date(year/month/day)
```

Suprtool checks the date's validity. To select the transactions for January 1999, you would do the following:

```
>item trans_date,date,phdate
>if trans_date >= $date(1999/01/01) and &
    trans_date <= $date(1999/01/31)
```

Relative Dates

Relative Dates

You can specify a relative date using the $date function.   Then you can create job streams that don't rely on hard-coded dates.   The general syntax of the $date function is:

$date(*year*/*month*/*day*)

The *year* can be a specific number (e.g., 2000) or an asterisk "*" for the current year.   To specify a relative year, you add or subtract years from the one you specified:

```
>if field=$date(2000/01/01)    {January 1, 2000}
>if field=$date(2000-1/01/01)  {January 1, 1999}
>if field=$date(*-1/01/01)     {January 1, last year}
```

The *month* can be a specific number (e.g., 6 for June) or an asterisk "*" for the current month.   To specify a relative month, you add or subtract months from the one you specified:

```
>if field=$date(2000/06-1/01)  {May 1, 2000}
>if field=$date(*/*/01)        {start of current year and month}
>if field=$date(*/*-1/01)      {start of last month}
>if field=$date(*/*-18/*)      {exactly eighteen months ago}
```

The *day* can be a specific number (e.g., 15), an asterisk "*" for the current day, the word "first" for the first day of the month, or the word "last" for the last day of the month.   You cannot add or subtract days to specify relative days; use $today instead.

```
>if field=$date(2001/01/first){January 1, 2001}
>if field=$date(*/*/*)        {today's date}
>if field=$date(*/*-1/last)   {last day of previous}
                              { month}
```

Combining these features should make it possible to generate batch jobs that require no operator input.   For example, to select all of the transactions for last month you would use:

```
>item trans_date,date,phdate
>if trans_date >= $date(*/*-1/first) and &
    trans_date <= $date(*/*-1/last)
```

Month End

Suprtool is always expecting a valid date.   Suppose that you have a month-end job that you run on May 31, 2000 and it contains the following If command:

```
>if field = $date(*/*-1/*)
```

If Suprtool used the literal interpretation of $date(*/*-1/*), it would use the date April 31, 2000.   In fact, there is no such date.   Whenever you specify * for the day and the day is greater than the last day of the month you specified, Suprtool uses the last day of the month instead of the current day of the month.   In our example, Suprtool would use April 30, 2000. Suprtool will take leap years into account when calculating the last day of February.

Today's Date

To select records based on today's date, use the following:

```
     >if field=$today          {today's date}
     >if field=$today(-1)      {yesterday's date}
     >if field=$today(+1)      {tomorrow's date}
```

Use the Item command to qualify the field as a date. The $today function accepts one optional argument which is the number of days before or after today. The maximum number of days in either direction is 9999.

yymmdd and Beyond 1999

Because dates beyond 1999 will not collate properly for the yymmdd form, you need to use $stddate to compare these dates.

```
>item ship-date,date,yymmdd
>if ship-date < $date(2000/12/31)             {will not work}
>if $stddate(ship-date) < $date(2000/12/31)   {will work}
```

Finding Invalid Dates

Use the $invalid function to find invalid dates. An invalid date is a number in a date format whose date equivalent cannot be found on a calendar. For example, a month value of 99 would be considered invalid.

```
>base store.demo
Database password [;]?
>get d-sales
>item deliv-date,date,ccyymmdd
>if $invalid(deliv-date)
>out baddates,link
>xeq
```

$Stddate Function

The $stddate function converts any date format in nearly any data-type container and internally converts it to the ccyymmdd format in a double integer container.

This allows you to compare dates with dissimilar formats and data-types.   For example,

```
     > in orddets
     > item order-date,date,ccyymmdd
     > item bill-date,date,mmddyyyy
     > if $stddate(bill-date) <= order-date
     > output badords,link
     > xeq
```

This feature is also available for dates that have two-digit years.   The century portion of the date will be generated by $stddate, which uses the normal cutoff rules specified by Set Date Cutoff.

```
     > in invdets
     > set date cutoff 20
     > item invoice-date,date,yymmdd
     > item close-date,date,mmddyyyy
     > if $stddate(close-date) <= $stddate(invoice-date)
     > out badinvs,link
     > xeq
```

In this case all invoice-date values with a *yy* portion between 20 and 99 will have a 19 for the century.   All invoice date values with a *yy* portion of less than 20 will have 20 for the date generated by the $stddate function.

Invalid Dates

A date must be valid before $stddate can convert it to the ccyymmdd format. Otherwise, a value of 0 will be returned for any invalid dates.   An invalid date is a number in a date format whose date equivalent cannot be found on a calendar.   This includes dates selected by the $invalid function.   We can eliminate the invalid dates from the above task by changing the If command slightly.

```
     > get invoice-detail
     > set date cutoff 20
     > item invoice-date,date,yymmdd
     > item close-date,date,mmddyyyy
     > if (not $invalid(close-date) &
       or  not $invalid(invoice-date)) &
       and     $stddate(close-date) <= $stddate(invoice-date)
     > out badinvs,link
     > xeq
```

In this example, if either the close-date or the invoice-date are invalid, then they will not be evaluated by the $stddate function and will not be selected.   Although your requirements may be different, you need to remember that invalid dates evaluated by the $stddate function will return a 0 value.

$Days Function

Suprtool supports a $days function, which converts any supported date to a Julian Day number (the number of days since 4713 BC). This allows for Date arithmetic, in which you can calculate the difference between two dates, even if they have dissimilar formats.

For example you could find all orders that were not shipped within 30 days of being ordered.

```
>form ordfile
    File: ORDFILE.SALES.MFG       (SD Version B.00.00)
       Entry:                  Offset
          ORDER-DATE        X8      1        <<CCYYMMDD>>
          SHIP-DATE         X8      9        <<MMDDYYYY>>
          ORDER-NUMBER      X6      17
    Limit: 10000  EOF: 15  Entry Length: 22  Blocking: 16

>in ordfile
>if $days(SHIP-DATE) - $days(ORDER-DATE) >=30
>list
>xeq
IN=15, OUT=4.  CPU-Sec=1.  Wall-Sec=1.
```

Invalid Dates

As with the $stddate function, if a date is not a valid date, then the result of the $days function will be zero. In the example above, if the order has not yet been shipped, then the SHIP-DATE will likely be blank, or zero, or some other special value. $Days(SHIP-DATE) will be zero, and the resulting calculation will be a negative number.

Notes on Relative Dates

The $date and $today functions always generate a constant from the date, just as if you had typed it. For example,

```
     >item field,date,yymmdd
     >if field > $today      {e.g., on February 13, 2001}
```

is the same as:

```
     >if field > 010213
```

Suprtool normally does no date conversion of the actual dates. Dates that do not start with the year do not collate correctly, so Suprtool does not allow relative comparisons with them (<, <=, >, and >=), although you may still compare for strict equality or inequality. The following examples will be rejected by Suprtool:

```
     >item trans_date,date,ddmmyy
     >if trans_date >= $date(*/*-1/first) and &
         trans_date <= $date(*/*-1/last)
        Error: Invalid date format for the comparison

     >input myfile,reclen 80, nolf
     >define mydate,1,6
     >item mydate,date,ddmmyy     {e.g., 301100}
     >define ....
     >if mydate > $date(00/11/01)
        Error: Invalid date format for the comparison
     >if mydate > $date(01/11/00)
        Error:  Invalid date:  Year = 1 Month = 11 Day = 00
```

If the date format does not allow the specification of a certain day, such as yymm, ccyymm, yyyymm, aamm, ccyy and mmyyyy, then you do not need to specify the entire date format, although Suprtool will allow either format for $date.

```
     >item trans_month,date,yymm
     >item purch_date,date,yymm
     >if trans_month <= $date(*/*/*) and &
         purch_date >= $date(00/01)
```

Because dates beyond 1999 in the yymmdd and yymm date types do not collate correctly, relative comparisons are no longer valid. Suprtool produces an error in the following case:

```
     >item trans-date,date,yymmdd
     >if trans-date >= $date(2001/01/01)
     Error: Cannot use a date beyond 1999 for this date format.
```

You can override this setting by entering the Set Date Ifyy2000error command:

```
     >set date ifyy2000error Off
     >item trans-date,date,yymmdd
     >if trans-date >= $date(2001/01/01)
```

Century

Oracle
Date limits
Workarounds

Century and $Date

Suprtool needs to generate a $date or $today date in the ccyymmdd format.   If you specify a two-digit year in the $date function, Suprtool needs to assume a century for the given date:

```
     >item trans-date,date,ccyymmdd
     >if trans-date >= $date(01/01/01)
```

Suprtool assumes 20 for the century if the specified year is less than the Set Date Cutoff value and 19 if the year is greater than this value.

Oracle Dates

Oracle dates contain both the date and the time.   The $date and $today functions check the date, but ignore the time.

Date Limits

The $date function in Suprtool can generate dates between the years 1583 and 2583. Some date formats have limits based on their particular format, such as 2027 for a Calendar date and 2259 for the aammdd aamm, mmddaa, ddmmaa dates.

Non-Collating Date Types

You can use the $stddate function to convert the non-collating date format to a J2 data item with a date format of ccyymmdd.

For example, to select the purchases by the field purch-date for November 2000 in a ddmmyy X6 field, you would use the $stddate function as follows:

```
>item purch-date,date,ddmmyy
>if $stddate(purch-date) >= $date(2000/11/first) and &
    $stddate(purch-date) <= $date(2000/11/last)
```

Dynamic Date Selection

You can use the If command for dynamic date selection.   Suppose you have a control file that maintains the start and end of a range of dates in which you are interested.   You can use the control file to select records from another file or dataset, based on this date range. This is a two-step process, in which the first Suprtool pass creates the If command with your dates, and the second pass does the actual selection from the dataset.

```
>input datecntl, reclen 12, nolf
>define start_date,1,6,byte
>define end_date,7,6,byte
>extract "if sales_date >= '"
>extract start_date
>extract "' and sales_date <= '"
>extract end_date
>extract "'"
>output seldate
>xeq
```

This produces a usefile that looks like

```
if sales_date >= '001101' and sales_date <= '001231'
```

Now you can use this file to do the actual selection:

```
>open oracle demo reader
>select * from sales
>use seldate
>output sdetail
>exit
```

Long Expressions

Long If commands can use an ampersand to continue the command over several lines.   This is awkward to use and, for internal reasons, the maximum length is restricted to 256 characters.   The $read function makes it easier to enter long If commands.   Its maximum length is based on the complexity of the expression, not on the number of characters.

Read Function
Redo
Data Overflow
Code Overflow
Usefiles

$Read Function

The $read function reads the If expression from $stdinx, or from the usefile if the If command is in a usefile. $Read continues to prompt for input lines until you press Return or enter "//."   You must remember to enter all the necessary parts of the If expression, including connectors like AND and OR and commas.   You do not use an ampersand (&) to continue from one line to the next when using $read.

```
>if $read                 {prompt for the expression}
-status = "20" and        {$read prompts with "-"}
-state = "AZ",            {the comma is still needed}
-        "CA",
-        "OR"             {no comma on the last line}
-                         {blank line to terminate $read}
```

Redoing $Read

When prompting for an expression, $read saves each line in the redo stack and accepts the Before, Do, Listredo, and Redo commands.   This provides an easy way to specify all or part of a previous $read expression.

Error: Data Overflow

While the $read function permits long expressions, there are other internal limits within the If command. The first is a limit on the amount of space for constants. Suprtool must blank-fill all string constants to their full length. The following example overflows the data space:

```
>define char-256,1,256
>if char-256 = "a","b","c"
Error:  Data Overflow
```

In this example, Suprtool attempted to create three 256-byte constants. There wasn't enough room for the last constant. Solutions to this problem include:

1. If possible, define short fields. If you have long field names, you may want to use the Define command to define shorter subfields.

2. Use tables and $lookup for many values.

3. Split the extract task into multiple passes. On the first pass, use an If expression that results in the fewest possible number of output records. Use the output file from the first pass as input to the second. Apply the remainder of your If expression during the second pass.

Error: Code Overflow

Suprtool translates If commands to an internal machine representation.   There is a limit on the size of this code.   When an error occurs, there is little you can do except use tables and $lookup wherever possible, and when this fails use multiple passes.

$Read in Usefiles

When you specify $read in a usefile, Suprtool expects the If expression to appear in the usefile.   This provides a method for storing and executing complicated If commands.

You can also manipulate Suprtool into prompting for portions of an If command. When the If command with $read is the last command in a usefile, Suprtool satisfies the $read from $stdinx.   The $read function can appear anywhere in which a space can appear, so you can use this to prompt the user for values.

Example
Notes

```
>use prompt.use
 >in  sdfile                    {first line of usefile}
 >if status=$read and &         {continue the If command}
     state = $read              {last line of the usefile}
 -"10"                          {prompt for status}
 -//                            {end of prompt for status}
 -"AZ",                         {prompt for state}
 -"CA",                         {user must remember comma}
 -"OR"                          {user also enters quotes}
 -//                            {end of the second read}
```

Notes

Suprtool is not designed to be used by end users.   We prefer that you write intelligent front-ends that understand user applications and hide the details of Suprtool from end users.   We recommend that you use $read from usefiles only for one-time tasks, or for tasks used by experienced Suprtool users who do not require a friendly user interface.

Input Command [I]

Opens the file that will be the input source for the next extract task. The file must contain fixed-length records with or without a line feed separator. If you are selecting records, the selection parameters must appear after any Reclen, LF, or NOLF parameters. The record length is the length of the data portion of each record, not including the line feed. If you specify LF, Suprtool/UX includes the line feed as part of the data by increasing the record length by one character.

If you have an input file with 80-byte records, and each record is separated with a line feed, you would use:

```
>input uxfile,reclen 80,lf
>list  char
>xeq
```

Since Uxfile has line feeds, the List command shows a dot (.) as the 81st character of each record. This dot corresponds to the line feed character. To read every fifth record in Uxfile, you would use:

```
>input uxfile,reclen 80,lf, (#5)
>list  char
>xeq
```

To examine the Suprtool object code (which has no line feeds between records), you would use:

```
>input /opt/robelle/bin/suprtool,rec 256,nolf
>num   10
>list  hex,char
>xeq
```

Suprtool executes the Input command immediately -- it does not wait for an Xeq command before opening the Input file.

INPUT *file* [,RECLEN *length*] [, LF | NOLF]
[(*startrecord*/[*endrecord*])]
[(#*count*)]

(Default: all input records)

File name
Record Numbers
Random Selection
Notes

Input File

The first example shows the most common use of the Input command.   An input file is specified as the input source to Suprtool.   We select a subset of the input data with the If command.   Before using the If command, we must define a field within the input record:

```
>input invent, r 80,nolf   {input is from a disc file}
>define a,11,2,int         {"A" is an integer that starts}
>output outfile            {    at the 11th byte of Invent}
>if a<10000                {records with field "A" less than}
>xeq                       {   10000 are written to Outfile}
```

Selection by Record Number

The (*startrecord/endrecord*) parameter specifies a range of input records by record number. The default value for endrecord is the highest record in the file.   The first record number is 0 for disc files.

The (*startrecord/endrecord*) parameter must come after the *reclen and LF/NOLF* parameters, if it is present.   You can check your record selections with the Verify command.

When debugging software, it is convenient to scan the first few records of a file.   The *startrecord/endrecord* parameter makes it easy to scan these records:

```
>input invent,r 80,nolf, (0/19)  {first twenty records}
>list                            {produce a list in }
>xeq                             {    "OCTAL,CHAR" format}
```

Random Selection

The (#*count*) parameter specifies that every "nth" record in the input file should be selected. This option is designed for random sampling of the input file.   The (*startrecord/endrecord*) parameter cannot be used with this parameter.   Like (*startrecord/endrecord*), (#*count)* must come after the *reclen, lf/nolf*, if present.

Test files can be constructed from random samplings of production files.   We can build a test file with the #*count* parameter:

```
     >input invent,r 80,nolf,(#15)  {every 15th record is read}
     >output dtest                  {create an output file with}
     >xeq                           {    every 15th record}
```

Notes

Only one Input or Select command is allowed per extract task.   The Input command opens the specified file *immediately*.   The file is held open until the input is reset or the current task completes.

Item Command   [IT]

Use the Item command to specify the number of implied decimal points or the date format for an item.   The $date and $today functions of the If command work only with dates.   The Item command must preceed any If or Extract commands.

```
        ITEM   itemname,DATE | DECIMAL,attribute
```

Itemname
Date Format
Decimal Places
Notes

Itemname

The *itemname* must be a Defined field, an SD field, or a column name.   The itemname cannot be qualified with a subscript.

Date Formats

For dates, the *attribute* must be one of the following:

ASK	ccyymm
Calendar	ccyy
ddmmyy	aammdd
ddmmyyyy	aamm
mmddyy	mmddaa
mmddyyyy	ddmmaa
Oracle	SRNChronos
PHdate	mmyyyy
yymm	yyddd
yymmdd	ccyyddd
yyyymm	HPCalendar
yyymmdd	JulianDay
yyyymmdd	EDSDate
ccyymmdd	PHDate8

Abbreviations
Data-Types
Limits
Calendar
Powerhouse
ASK
yyymmdd
EDSdate
Julianday
aammdd
Oracle
SRN Chronos
ddd
HPCalendar

Abbreviations

When specifying the Date keyword, you can use a leading subset for the date attribute.   For example, if you want to specify the Calendar date type, you can specify only CA.

```
     >item cal-date,date,ca
```

If you do not like this feature, you can turn it off by specifying the following command in your Suprmgr file:

```
     >set itemabbreviatedate off
```

Data-Types for Dates

Each date *attribute* is compatible with certain data-types. For more information, see the table on data-types in the Define command. The following table shows the compatibilities:

Date-Attribute	Data-Type Compatibility
ASK	J1 and K1
Calendar	J1 and K1
ddmmyy	X6, Z6, J2, K2, and P8 or greater
ddmmyyyy	X8, Z8, J2, K2, and P10 or greater
mmddyy	X6, Z6, J2, K2, and P8 or greater
mmddyyyy	X8, Z8, J2, K2, and P10 or greater
Oracle	X7
PHdate	J1, K1, J2, and K2
yymm	X4, Z4, J1, and K1
yymmdd	X6, Z6, J2, K2, and P8 or greater
yyymmdd	J2, P8
yyyymmdd	X8, Z8, J2, K2, and P10 or greater
ccyymmdd	X8, Z8, J2, K2, and P10 or greater
ccyymm	X6, Z6, J2, K2, and P8 or greater
yyyymm	X6, Z6, J2, K2, and P8 or greater
aammdd	X6
aamm	X4
mmddaa	X6
ddmmaa	X6
ccyy	X4, Z4, J1, and K1
SRNChronos	X6
mmyyyy	X6, Z6, J2, K2, and P8 or greater
yyddd	X5, Z5, J2, K2, and P8 or greater
ccyyddd	X7, Z7, J2, K2, and P10 or greater
HPCalendar	J2, K2
EDSDate	J2, P8
JulianDay	J2
PHdate8	J1, K1, J2, and K2

Date Limits

The $date function in Suprtool can generate dates between the years 1583 and 2583. Some date formats have limits based on their particular format, such as 2027 for a Calendar date and 2259 for the aammdd aamm, mmddaa, ddmmaa dates.

Calendar

The Calendar *attribute* is provided for users who have fields containing the 16-bit MPE Calendar date format as an unsigned, **logical** value with seven bits for the year of the century (bits 0-6), followed by nine bits for the day of the year (bits 7-15).   The Calendar date format only supports dates up to the end of the year 2027.

PHdate and PHdate8

The PHdate and PHdate8 *attributes* are compatible with the COGNOS PowerHouse date format.   If the *data-type* is J1 or K1, the date is stored as a LOGICAL value with seven bits for the year of the century (bits 0-6), four bits for the month (bits 7-10), and five bits for the day (bits 11-15).   If the *data-type* is J2 or K2, the date is stored as yyyymmdd.

PHDate and PHDate8 date formats are similar, however PHDate values for the year range from 0 - 99, whereas PHDate8 year values are from 0 - 127.   A year of 0 in PHDate could mean either 1900 or 2000 depending on user applications. A year of 0 in PHDate8 means 1900, and 100 means 2000.   The PHDate8 date format is found in PowerHouse version 8.19 and higher.

ASK

The ASK *attribute* is compatible with the ASK manufacturing software. ASK uses a special date format stored as a single integer or a single logical (i.e., J1 or K1 in IMAGE). This date is relative to January 1, 1973.

yyymmdd

The yyymmdd *attribute* is similar to yymmdd, except that the first digit denotes the century. If the first digit is a 1 (one) then the century is 19, and if the first digit is a 2 (two) then the century is 20.   Only data-types of P8 and J2 are supported for this date *attribute.*

This date format is used by some third-party software packages such as MACS and APS.

EDSDATE

The EDSDATE date format is similar to the yyymmdd format, in which the first digit represents the century.   The first digit in the EDSDATE is either 0 or 1: a 0 represents a century of 19 and a 1 represents a century of 20.

JulianDay

The JulianDay number is the absolute count of the days that have elapsed since January 1, 4713 BC on the Julian calendar.

Typically "Julian Day numbers" refer to an integer number corresponding to whole days, while the "Julian Date" may mean an integer plus a decimal value that resolves the Julian count to precise parts of a day.   Suprtool supports the "JulianDay" number and does not attempt to support an hour or point in the day.

aammdd and Related Date Formats

The aammdd *attribute* is similar to yymmdd, except the aa portion of the date uses a combination of letters and numbers in order to represent dates beyond 1999.

The aammdd date format was developed by James Overman of HP for use in their MM3000 product. This format is only available for X6 data-type.

By substituting a letter of the alphabet in the first position of the year, we can extend a six-digit date and also ensure that the dates collate correctly. For example:

YY of AAMMDD	**CCYY**
A0 - A9	2000 - 2009
B0 - B9	2010 - 2019
C0 - C9	2020 - 2029

Because letters are greater than numbers in the collating sequence you can ensure that aammdd dates beyond 1999 will order correctly.

Suprtool also supports other date formats with this two-digit year representation. These formats are aamm, mmddaa and ddmmaa.

Oracle

Oracle dates include both the date and the time.   The $date and $today functions only apply to the date part of Oracle dates.

SRN Chronos

The Srnchronos date format is used in applications from Software Research Northwest.   Like Oracle dates, this format includes the date and time, but the time portion is ignored by the $date and $today functions.

DDD Dates

Dates consisting of ddd in the format name use the ddd to represent the *n*th day in the current year.   This means that January 1 will be day 001, and Dec 31 will be day 365 on non-leap years.   Some people refer to these type of dates as Julian dates.

HPCalendar

The HPCalendar date format is supported by HP's new HPCalendar intrinsic and consists of a 32-bit integer number, whereby the first 23 bits represent the year and the last nine bits represent the day of the year.

Decimal Places

The decimal *attribute* is the number of implied decimal places in an item. The minimum number of implied decimal places is 0. The maximum is based on the data-type of the item:

Data-Type	Maximum Implied Decimal Places
I1	5
I2	10
I3	15
I4	19
K1	5
K2	10
P*n*	*n-1*
Z*n*	*n*

You cannot specify implied decimal places for byte-, char-, or IEEE-type items.

Once you define a decimal place, almost every command in Suprtool is affected. Suprtool accepts numeric values with decimal points or scales integers according to the number of implied decimal places (e.g., specify two implied decimal places, then enter 1,000 to represent 1,000.00). All formatting commands format fields with a decimal point when appropriate.

Constants

Constant Values

When specifying numeric constants for a field with implied decimal places, there are different formats that you can use.   For example, assume that we use the Item command to specify two implied decimal places for an amount field. The following are examples of constant values for this item:

`0`	{zero value padded as necessary}
`1`	{$1.00}
`0.01`	{$0.01}
`.01`	{also acceptable for $0.01}

Notes

SQL columns
Compound Items
When to Use
Usefiles

SQL Columns

You must redefine any SQL columns before you can use the Item command.

```
>sel  * from emp
>def  salary,sal
>item salary,decimal,3          {correct scale}
>if salary > 15.275
```

Compound Items

When you specify a compound item, the attribute applies to all elements of the compound item.

```
>item monthly_totals,decimal,2          {12 occurrences}
>if monthly_totals(5) > 1000.00
```

You cannot apply an attribute to only one sub-item of a compound item:

```
>item monthly_totals(5),decimal,2

Error:  Missing attribute for the Item Command
```

When to Specify the Item Command

The Item command affects almost all other Suprtool commands.   It should be used as follows:

¤ For databases, specify the Define and Item commands immediately after the Select command, but before any Extract or If commands.

¤ For disc files, it is best to specify the Define and Item commands immediately after the Input command.

Usefiles

You can use a usefile as a mini data dictionary for Suprtool.   For disc files, you can put both the Define and Item commands in a usefile and execute them right after specifying the file with the Input command.   If you use the Link output option, both date formats and implied decimal places are saved in the self-describing file so they never need to be specified again.

Key Command   [K]

Specifies the next sort field for an extract task, using an explicit byte position in the record, not a field name.   See the Sort command for specifying sort fields by table column name or Defined field, or by field name in an SD file.   Up to 20 Sort and Key commands may be specified per extract task; the first is the major sort field.

KEY   *byteposition*,*bytelen* [,*type*] [,DESC]

(Default: *type*=BYTE, ASCENDING order).

Parms
Examples
Notes

Parameters

Desc specifies that the field is to be sorted in Descending order.

The *byteposition* is a number between 1 and the length of the input records, specifying where the sort field begins. The *bytelen* parameter is a number from 1 to the length of the record, specifying how long the sort field is.

The *type* is a word that gives the desired data-type of the field for sorting purposes:

BYTE	(unsigned, straight compare)
INT/INTEGER	(two's complement)
DOUBLE	(two's complement)
IEEE	(IEEE floating-point)
PACKED	(packed-decimal)
PACKED*	(packed-decimal, last nibble unused)
DISPLAY	(zoned-decimal numeric field)
LOGICAL	(unsigned, like BYTE, 2 characters)
CHARACTER	(for Native Language Support)

The Key command also accepts FPOINT as the data-type for IEEE numbers.

Examples

The first example sorts an integer (PIC S9(4) COMP) field which starts on the 11th byte of the input record.   We sort the entire input file based on one key:

```
>input bigfile,r 40,lf   {input from a disc file}
>key 11,2,int            {key is an integer that starts}
>output outfile          {    at the 11th byte of Bigfile}
>xeq                     {create Outfile and prompt for}
                         {    more Suprtool commands}

>input discfile,r 40,lf  {another input file}
>key 14,4,double,desc    {double integer (PIC S9(9) COMP)}
>output ofile2           {sort input in descending order}
>exit
```

The following examples show the various data-types and combinations that are available with the Key command:

```
>key 1,10              {byte data-type}
>key 1,10,desc         {descending sort sequence}
>key 11,4,double       {I2 or J2, S9(9) COMP}
>key 1,6;11,4,ieee     {X6 string and an E2 field at byte 11}
>key 21,6,packed       {P12, S9(11) COMP-3}
>key 21,6,packed*      {P12, S9(10) COMP-3, wasted byte}
```

Notes

The command name Key is optional.   Any command that starts with a numeric character is assumed to be a Key command.   The Verify command shows all of the current key fields, and the Reset command cancels them.   If no sort fields are defined prior to the Xeq or Exit command, Suprtool performs a copy, not a sort.

Link Command   [LIN]

You cannot use Suprtool's Link command to invoke Suprlink/UX, but you can run Suprlink/UX by itself.

```
   /opt/robelle/bin/suprlink

   Suprlink/UX/Copyright Robelle Solutions Technology Inc.  1988-2001
   (Version 4.4.13)
   +
```

Suprlink/UX provides high-speed data file linking based on a sort key. Suprlink/UX only accepts self-describing files created by Suprtool/UX or the MPE SDUnix program.

List Command [L]

The List command is used to produce formatted listings of the selected records. You may specify the List command, or the Output command, or both, or neither. If List is used instead of Output, Suprtool sets the Output to $null, so that only a listing is produced.

```
LIST    [ OCTAL|HEX|DECIMAL ] [ CHAR ] [ NOREC ] [ LABELS ]
        [ RECORD ] [ DUPLEX ] [ ONEPERLINE ] [ LP ]
        [ NONAME ] [ NOSKIP ] [ STANDARD ] [ DEVICE name ]
        [ DATE format ] [ TIME format ] [ PCL format ]
        [ LEFTJUSTNUM ] [ RIGHTJUSTNUM ]
        [ TITLE "string" ]
        [ HEADING "string" ["string" ...]]
```

(Default: Octal/Char or "Formatted")

If Suprtool knows about the fields in the input source (e.g., because you have used the Extract command), the list records are formatted with field names, and internal binary data-types (e.g., integer) are converted to ASCII. You cannot combine the Ask or Query,Num output-options with the List command.

Here is a typical use of List: to find any entries in the Customer table that do not have a valid value for "status".

```
>open oracle demo reader       {input from a database}
>select * from customer        {read this table}
>if status<>10,20,30,40        {the only valid values}
>list                          {print bad entries}
>xeq
```

Format
Laserjet
Headings
Simple Reports
List Device
Notes

Format

You can override the defaults with a specification in the List command (e.g., List Hex,Char). If the input source is not self-describing and no Extract command is specified, the default output format is Octal,Char, which also shows both input and output record numbers.

Decimal Places
Record Numbers
One Per Line
No Field Names
Blank Lines
Numeric Justification

Decimal Places

The List command formats numbers using the implied number of decimal places. For example, the following Suprtool commands format the unit cost with two decimal points. We specify the Rightjustnum keyword, since numbers with decimal points are hard to read if they are left justified:

```
>item cost,decimal,2           {two implied decimal points}
>list rightjustnum             {numbers right justified}
>xeq                           {   with decimal points}
```

Listing Record Numbers

The Norec keyword prevents the printing of the input and output record numbers.   The input record numbers are not printed if Output xxx,Data is used and the file is sorted.

Listing One Field per Line

Suprtool normally attempts to list more than one field on every line of list output.   The Oneperline keyword causes every field to be shown on a different line.

Listing without Field Names (Noname)

When Suprtool knows the record structure of the output file, it shows the name of each output field. The Noname keyword causes the field names to be suppressed. By only extracting a few fields, it is possible to fit the listed output for each record on one line.

Suppressing Blank Lines Between Records

By default, Suprtool prints a blank line between each record. The Noskip keyword removes this blank line. If you combine the Noskip, Norec, Noname, and Title options when extracting a few fields, Suprtool can produce a simple report.

Numeric Justification (Leftjustnum and Rightjustnum)

The List command normally left justifies all numeric fields. Specifying List Standard causes all numeric fields to be right-justified, unless you override the default with the Leftjustnum keyword.

Use the Leftjustnum or Rightjustnum keyword to specify the alignment of the numbers. The two keywords are mutually exclusive. The last one that appears on the command line is the one that is applied.

LaserJet Listings

There are two methods to select different printing options for a LaserJet and other PCL-compatible printers. You can permanently set the PCL option for all listings by using Set List PCL, or you can use the List command to select the PCL option for just one task. PCL stands for Printer Command Language, which is an HP standard for printers. The following is a summary of the PCL values:

1	Lineprinter	landscape	175 cols/60 lines
2	Courier	landscape	100 cols/45 lines
3	Courier "standard"	portrait	80 cols/60 lines
4	Lineprinter	portrait	132 cols/80 lines
5	Courier A4 "tight"	portrait	80 cols/60 lines
6	Lineprinter legal	landscape	223 cols/60 lines

See the Set command for a complete description of the PCL options. By default, Suprtool assumes that the List output device is not PCL-compatible (List PCL 0).

If you use the List command to your terminal with a global Set List PCL value other than zero, your terminal screen may be cleared. To avoid this situation, you can explicitly specify the PCL setting along with the device:

```
>get d-sales
>list serialp,pcl 2
>xeq
```

A4 Paper
Ascii
Double Sided

A4-Size Paper

Most of the PCL options, with the exception of PCL 5, were designed and tested with North American letter-size paper.   This is especially true with PCL 5 for A4 paper: it reduces the horizontal spacing between characters so that 80 columns of Courier output fits on a single line.   In addition, if you add 2000 to a PCL code, Suprtool adjusts the number of rows and columns for that option to match A4 paper.   For example, to print Landscape on A4 paper, use PCL 2001 instead of PCL 1.

In general, selecting A4 paper gives you more space along the long side of the paper and less space along the short side.   If you are happy with the way letter-size rows and columns work on A4 paper, simply do not add 2000 to the PCL code.

Roman-8 vs. ASCII

The PCL option requests a Roman-8 character set, but some combination font cartridges only supply the ASCII character set (half as many characters means twice as many fonts in a single cartridge).   If you ask for Landscape Lineprinter and get Landscape Courier instead, your Lineprinter font probably has the ASCII character set instead of the Roman-8 character set.   To request an ASCII font, add 1000 to the PCL code.   For example, if you have a Super Cartridge (55 fonts in one!), use PCL 1001, 1004, and 1006.   To select both ASCII and A4 paper, add 3000.

Double-Sided Printing on LaserJets

The LaserJet IID and IIID can print on both sides of the paper.   The Duplex keyword enables double-sided printing on these printers.

```
     >list duplex
```

Headings in Listings

Specifying a title in the List command forces Suprtool to produce a formatted listing with page-headings, page-numbers, today's date and the current time. If you want just the date and page numbers, use an empty string (e.g., `List title " "`). The following example prints a report on a LaserJet in Landscape (sideways) mode, using the tiny Lineprinter font, including a page heading with the title. The physical command line limit is 256 characters. As a result, the maximum size of the heading is less than 256 characters because the List command and heading options need to be included in the command line.

```
>in custs                    {self-describing file}
>if status<>10,20,30,40      {the only valid values}
>set list pcl 1              {select LaserJet option}
>list title "Invalid CUSTOMER Records"
>xeq                         {include title on listing}
```

Date Format
Time Format

Changing the Date Format

When you select page headings by specifying a title, each page includes today's date. By default, this date is formatted as mmm dd, ccyy (e.g., Mar 20, 2000). You can override this format with the Date keyword. Use the Set command to specify a different default date format for future List commands (e.g., Set List Date 2). The valid date formats are as follows:

0	mmm dd, ccyy	Mar 20, 2000 (default)
1	yy/mm/dd	00/03/20
2	mm/dd/yy	03/20/00
3	dd/mm/yy	20/03/00
4	dd mmmyy	20 Mar00

```
>list title "Example Report" date 3
>xeq                 {heading date is in dd/mm/yy format}
```

Changing the Time Format

When you select page headings by specifying a title, each page includes the current time. By default, the time is in 24-hour format (e.g., 23:02). You can override this format with the Time keyword. Use the Set command to specify a different default time format for future List commands (e.g., Set List Time 2). The valid time formats are as follows:

0	none		
1	24-hour	23:02	(default)
2	AM/PM	11:02PM	

```
>list title "Example Report" time 2
>xeq                    {time will be in AM/PM format}
```

Simple Reports

A Fast Method for Producing Simple Reports

For self-describing files and database tables, the Standard keyword is equivalent to List Noname,Noskip,Norec,Rightjustnum with default column headings. For data files, the Standard keyword is equivalent to List Octal,Char. In either case, the Standard keyword provides a default title that describes the input source. You can override the title, date format, time format, or any other option selected by the Standard keyword, by specifying them in addition to the Standard keyword. For example,

```
      >select * from customer
      >list standard              {use all Suprtool defaults}
      >xeq

      >select * from customer
      >list standard,date 3       {override the date format}
      >xeq

      >input uxfile,reclen 80,nolf
      >list  standard,char        {override the format options}
      >xeq

      >input uxfile,reclen 80,nolf
      >list  standard,leftjustnum {left justify numbers}
      >xeq

      >select * from customer
      >list standard,title "Customer List"
      >xeq                        {override title}

      >select * from customer
      >list standard,heading " "  {no column headings}
      >xeq
```

Subheadings

Listings with Subheadings

When using the Title or Standard keywords, you can also include subheadings with the Heading keyword.   You can specify multiple columns by repeating the string after the Heading option (e.g., List Heading "First " "Second") or specify the Heading option multiple times (e.g., List Heading "First ", Heading "Second").   Being able to specify multiple columns makes it easier to align column headings when using the Standard keyword.   If you specify the Heading keyword without the Title keyword, a default title is produced.

```
>select  * from customer     {read this table}
>extract account             {Z8 }
>extract firstname           {X10}
>extract lastname            {X16}
>extract rating              {J2 }
>extract status              {X2 }
>sort lastname
>sort firstname
>list standard,title "Customer List", &
      heading "Account  " " First and Last Names" &
                "        " "Credit   " "Status"
>xeq
```

List Device

By default, the List command lists lines to $stdlist.   There are several methods that you can use to redirect the output to a specific logical device or to a disc file.   You can also redirect output to an attached printer.

Lp
User Specified
Attached Printer

Device LP

Use the LP option to send listings to the lp command.   The LP option assumes the list device is 80 columns wide.

```
>list lp
```

User Specified Device

Use the Device option to specify a specific logical device for the listing. The device name must appear after the Device option.   The device name is case-sensitive so that the device "SHIPPING" is different than the device "shipping".   The Device option assumes that the list device is 80 columns wide.

```
     >list device LP         {same as List LP}
     >list device serialp    {send to device serialp}
```

Listing to Attached Printer

If you wish to list to a printer that is attached to your terminal, use List Record. Suprtool uses Record mode on your terminal or PC to print on the attached printer. To use this option, you must be using an HP terminal or HP terminal emulator and your data communication settings must be setup correctly.

You can combine this option with other listing options. You cannot interrupt Record mode with Control-Y, but you can do a Soft Reset. This unlocks the keyboard and causes the rest of the output to appear on the screen. You can then stop it with Control-Y.

List record may not return complete control to your PC when running Reflection. The report printed and the keyboard unlocks, but control is not passed to your terminal. You can get control back by doing a Soft Reset (Alt-S). You can prevent this problem by setting DISABLE-COMP-CODES to yes.

If you are listing to an attached printer from a terminal, your terminal may remain locked after the printout is completed. This generally happens when you have handshaking enabled. (G-H straps set to No). You can do a soft reset to unlock your terminal.

If handshaking is disabled (G-H straps set to Yes), the List command works and returns control to the terminal but two "S" characters will be printed on the terminal. There is currently no known workaround to these problems.

Notes

The List operation occurs logically after the sort phase, if any, and after any Extract, but before the final Output or Put operation.   A Reset turns off the current List options.

For more examples of the List command, see "Suprtool Issues and Solutions."

Listredo Command [LISTREDO]

The Listredo command displays any of the previous 1000 commands.

```
LISTREDO     [ start [ / stop ] ]          [;ABS] [;OUT=file]
             [ string ]                    [;REL]
             [ ALL | @ ]                   [;UNN]
```

(Default: display previous 20 commands)
(BJ and ,, are short for LISTREDO)

Commands are numbered sequentially from 1 as entered and the last 1000 are retained. You can display a single command, a range of commands, all 1000, or all the commands whose name matches the string. You can display the commands with ABSolute line numbers (the default), RELative line numbers (-5/-4), or UNNumbered. If you want to redo any of these commands, see Do, Redo, and Before.

Examples
Saving
Notes
Persistence

Examples

```
>listredo 5
>listredo 5/10
>listredo help      {print all Help commands}
>listredo -10       {print last ten commands}
>listredo ALL       {print entire redo stack}
>listredo rm        {print all rm commands}
>listredo rm xx     {print all "rm xx" commands}
>listredo @rm      {print all with "rm" anywhere}
>listredo @;rel    {print ALL, relative numbers}
```

Saving to a File

Saving the Listredo commands to a file is not supported in Suprtool/UX.

Notes

The :Listredo command can be abbreviated to BJ as in Qedit, or to ,, as in MPEX. You cannot use a semi-colon (;) to combine commands on the same line.

Persistent Redo

Redo commands can be saved in a permanent file and can therefore be used from another session.   You can use the **Set redo** command to specify a filename to save your redo commands.   Please see the Set Redo command for details.

Numrecs Command   [N]

Limits the number of records selected and the size of the sort scratch file.

```
NUMRECS   size

          or

NUMRECS   percentage%
```

(default: *size*=10,000 or EOF of input source)

Parameters
Reducing File Size

Parameters

To limit the number of records selected from the input source and to reduce the size of the sort scratch file, use the Numrecs command. If you select more than *size* entries, Suprtool prints a warning message, and ignores the rest of the input records. However, the output file will have the records that were selected. Use a percent sign (%) to specify the Numrecs as a percentage of the input file size. The percentage can range from 1 to 500, but values over 100 have no effect on HP-UX.

Reducing File Sizes

Suppose that the Sales table contains 100,000 entries, but you use the If command to select 15% of the entries.   We would specify 15 as the *percentage* on the Numrecs command to reduce the size of the sort scratch file and the output file:

```
>select * from sales   {specify input}
>numrecs 15%           {specify 15000 as file size}
>if qty<100            {select a subset of Sales}
>sort account          {sort by customer account}
>output out2           {output file will have room for}
>xeq                   {      15000 records}
```

Open Command   [OP]

Specify an SQL database to open.   Only one database can be open at a time.

```
        OPEN   ORACLE   *username [password]*
        OPEN   ALLBASE *dbename   owner*
```

Oracle
Allbase

Oracle Database

Use the Open command to connect to an Oracle database.   You must specify your Oracle *username* and *password*.   If you do not specify the password in the Open command, Suprtool prompts for your password (without echoing it on the screen).   To read data from an Oracle table or view, use the Select command. Use Set Oracle Rows to specify how many rows for Suprtool to fetch when it is reading Oracle data.

```
    >open oracle scott tiger
    >open allbase inventory anne
```

Allbase Database

Use the Open command to connect to an Allbase database. You must specify your Allbase *dbename* and *owner*. To read data from an Allbase table or view, use the Select command. Use Set Allbase Rows to specify how many rows for Suprtool to fetch when it is reading Allbase data.

```
    >open allbase inventory anne
```

Output Command [O]

Define the name of the output file as one of these: a new disc file (default), an existing file (Append or Erase option), or * for $stdlist. If you use List or Total, Output defaults to $null. Output produces the same record format as the input source (adjusted by Extract commands), unless you override it with format keywords.

```
OUTPUT   filename format [ERASE | APPEND]
```

(Default: DATA only, "new" file)

New
Old
Stdlist
Format
Examples
Notes

New Files

The *filename* is the name of a new disc file to be built by Suprtool (Output xxx).   If Suprtool cannot Save the new file because of a duplicate file name, you may purge the old file or give the new file a different name.

If Suprtool is running in batch mode, it renames the new file as Output*nn*, where *nn* is the lowest number between 00 and 20 that makes a valid new file name.   The Output*nn* file is built in the same directory as the duplicate output file and not in your current directory.

Existing Files

The *filename* is the name of an old disc to erase (Output xxx,erase) or to append to (Output xxx,append).   Output xxx,erase does not purge the existing file; it simply overwrites the contents of the file.

$Stdlist

Specify output to $stdlist by a single asterisk (Output *).   Use this *filename* with the ASCII option for a quick listing of a file or database table.

Format Options

The format and length of the output records is determined by which of the following *format* keywords is selected:

DATA	Default
KEY	Sort keys only
NUM	J2 record numbers only
NUM,KEY	J2 plus sort keys
NUM,DATA	J2 plus data record
QUERY	Self-describing file
LINK	New format self-describing file
NUM,QUERY	Query "numbers" format
ASK	COGELOG ASK select file
ASCII	Convert numeric to ASCII
DISPLAY	Convert numeric to display (zoned-decimal)
PRN	Personal computer format
NOLF	Do not write out Line Feeds to the end of the record
LF	Write out Line Feeds to the end of the record

Data
Key
Num
Combinations
Query
Ask
Ascii
Display
Prn
NOLF
LF

DATA.

DATA is the default; output records are the same length and format as the input records.

KEY.

KEY creates output records containing only the sort keys, concatenated from left to right (major key to minor key).

NUM.

NUM creates output records of four bytes in length, containing the double-integer record number of each input record.   The record number of the first record is 1 for database tables and 0 for disc files.

NUM,KEY.

NUM,KEY creates output records containing the four-byte record number, followed by the sort key values, concatenated from left to right (major key to minor key).

NUM,DATA.

NUM,DATA creates output records containing the four-byte record number, followed by the full data record.   This can be useful to create "transaction" files from detail datasets that will later be updated back to the database after a processing stage.

QUERY.

QUERY creates a self-describing output file.   A self-describing file is a data file plus an extra file which contains information about the structure of each record in the data file.   This extra file is created with the output file name plus the extension ".sd".   This ".sd" file contains the name, type, length, and offset of each field in the data file.

QUERY self-describing files have no provision for repeated fields.   These fields appear with an   "unknown" type in the .sd file.   See the LINK option for a better self-describing file format.   The QUERY option produces a file that is similar to the file produced by the SAVE command in the MPE QUERY program.

Link
Combinations

LINK.

The QUERY output option has some major drawbacks.   Compound fields are not handled, date and decimal point information is not saved, and sort information is not part of the file description.   The LINK option produces a self-describing file that solves all these problems. This option is the recommended way to generate files for Suprlink.   We are also encouraging third-party software vendors to accept this format.

To convert a self-describing file back into a non-SD file, simply purge the corresponding ".sd" file.

NUM,QUERY.

NUM,QUERY creates output records in QUERY "numbers" format. This is an undocumented feature of QUERY that is used by some application packages. Records in a "numbers" file contain the dataset number and the record number in ASCII format. A QUERY "numbers" file is usually used as input to a report program. Substituting Suprtool for QUERY in these batch jobs should improve the speed.

ASK.

ASK is a QUERY-replacement tool from COGELOG.   ASK creates output records in ASK select-file format.   Suprtool can be used to scan and select records quickly from a dataset.   ASK Version C can produce reports from the resulting select-file.   This option cannot be used with any other Output option.
Recent versions of ASK read self-describing files.   For these versions use the Link option instead of the Ask option.

ASCII.

ASCII converts all binary input fields into their equivalent ASCII values. All binary fields are right-justified in their fields with a trailing sign. The sign can be a blank, "+", or "-". Byte fields are not affected by this option. The size of the ASCII field depends on the format of the binary field:

I1 J1	06 bytes	I2 J2	11 bytes
I3 J3	16 bytes	I4 J4	20 bytes
K1	05 bytes	K2	10 bytes
E2	12 bytes	E4	23 bytes
Z*n*	*n*+1 bytes	P*n*	*n* bytes

Any fields with implied decimal places (see the Item command) are formatted with a decimal point in the correct position. Suprtool reserves two additional positions for each output field that has an implied decimal point.

DISPLAY.

DISPLAY converts all binary input fields into their equivalent display values (also known as zoned-decimal). The size of the display field depends on the format of the binary field:

I1 J1	05 bytes	I2 J2	10 bytes
I3 J3	15 bytes	I4 J4	19 bytes
K1	05 bytes	K2	10 bytes
P*n*	*n*-1 bytes	P**n*	*n*-2 bytes

The last digit of the number is replaced with a letter corresponding to positive or negative values. See the following table.

Units Digit	Internal Representation (ASCII)		
	Positive	Negative	No Sign
0	{	}	0
1	A	J	1
2	B	K	2
3	C	L	3
4	D	M	4
5	E	N	5
6	F	O	6
7	G	P	7
8	H	Q	8
9	I	R	9

Positive values for I- and J-type fields are converted into neutral (i.e., no sign) display values. The sign is preserved when converting packed fields to display.

DISPLAY is not supported for Real or IEEE fields. Byte fields are not affected by this option. Display fields in the input record are not converted into display.

PRN.

Many PC software packages import PRN files (these files are also called delimited in some PC documentation). A PRN file has quotes around character-fields and a comma between each field. Binary values are output in ASCII with an optional leading sign. Not all applications accept PRN files; for more precise conversion of data, use STExport.

PRN converts each input-field to a fixed-width output-field, filling with trailing spaces where necessary. All binary values are converted into their equivalent ASCII value, left-justified in their fields. The sign precedes the ASCII value for the number and can be missing, "+", or "-". See the ASCII output-option for the field width of each data-type.

Output fields with implied decimal places (see the Item command) are formatted with a decimal point in the correct position. Like the ASCII option, Suprtool reserves two extra columns for each output field with implied decimal places.

Lotus 1-2-3 accepts records only up to 240-bytes long. Because the PRN option leaves room for the maximum value of any field, you may need to restrict the number of output fields using the Extract command.

While some PC software allows alternate characters to be used to delimit character fields, Lotus 1-2-3 accepts double quotes only. Since Lotus 1-2-3 rejects character fields that contains a double quote, Suprtool *removes* all double quotes from character fields when generating the PRN format. Suprtool removes quotes by replacing them with a space.

See "Suprtool Issues and Solutions" for a method of including header lines in the file to be down-loaded.

NOLF.

If you need to ensure that line feeds are not written to the end of each record, then you should specify the NOLF option.   It is usually preferable to specify the LF option.

LF.

If you need to ensure that line feeds are written to the end of each record, then you should specify the LF option.   When extracting from SQL databases, Suprtool writes out records without line feeds.   Files with line feeds are usually processed more easily by most other applications or import programs, so it is advisable to use the LF option if you are uncertain.

By default, whether Suprtool writes out a line feed depends on the input source.   For example, if the input source has line feeds, then line feeds will be written out at the end of each record.

When filling up PowerHouse subfiles, some versions of Quiz will abort if no line feeds are found at the end of the record.   It is recommended that when you write to a PowerHouse subfile, you should always use the LF option on the Output command.

Examples

One reason to use $stdlist as the output file is to obtain a quick listing of the ASCII fields in the input source.   The following example lists the Account, Lastname, and Firstname columns of the Customer table and separates them by two spaces:

```
>open ora demo reader {input from a database}
>sel  * from customer {use the Customer table}
>extract account      {account number will be first}
>extract "  "         {two spaces}
>extract lastname     {the customer's last name}
>extract "  "         {two more spaces}
>extract firstname    {the customer's first name}
>output *             {output the records to $stdlist}
>sort account         {sorted by the account number}
>exit
```

The following examples demonstrate other combinations of the Output command. The entire *Issues* chapter of the manual should be reviewed for extended examples using the Output command.   Many Output options were intended for specific application areas.

```
>output newfile
>output accum,append
>output keyfile,key
>output transf,num,data
>output querynum,num,query
>output $null
```

If you want to find out how many records are qualified by some selection criteria, without producing an output file, send the output to $null.   The Out= count on $stdlist displays the number of qualifying records.

Notes

The output file can *only* be an "old" file if the Append or Erase option has been specified.

Put Command [P]

The Put command is not available in Suprtool/UX.

Q Command   [Q]

Prints a message on $stdlist.

```
        Q    [ string ]
```

(Default: print a blank line)

Parms
Examples

The *string* of up to 253 characters is printed on $stdlist.   The string is truncated to the record width of $stdlist.

The string must be embedded in quotes.   Either single-quotes (') or double-quotes (") are permitted.   The quotes are not printed on $stdlist.

Examples

The Q command is often combined with the Userpause command.   The example is a usefile that provides an explanation of how long a task takes:

```
>q
>q "We will select all transactions over $10,000.  Since"
>q "there are many transactions, this task will take"
>q "some time (usually more than fifteen minutes)."
>q
>userpause "Press any key when you are ready to start."
```

Redo Command   [REDO]

Enables you to modify and repeat any of the previous 1000 command lines.

```
REDO    [ start [ / stop ] ]
        [ string ]
        [ ALL | @ ]
```

(Default: redo the previous command)
(Comma "," is short for REDO)

The Redo command allows you to modify the commands before it executes them. If you don't need to change them, use the Do command.   Commands are numbered sequentially from 1 as entered and the last 1000 are retained.   Use the :Listredo command to display the previous commands.   You can redo a single command, a range of commands, or the most recent command whose name matches a string.

The Redo command uses MPE-style editing logic (D, I, R, U and >).   The default mode is to replace characters.   To delete, type DDDD under the characters to be removed.   To insert, type I under the insertion spot, then the new characters.   To undo your changes, type U.   To append to the end of the line, use >xxx.   To delete from the end of the line, use >DD.   To replace at the end of the line, use >Rxxx.   And to erase the rest of the line, use D>.   If you prefer Qedit-style editing (Control-D, etc.), use the Before command instead of the Redo command.

Examples
Notes
Editing
Persistence

Examples

```
>ll *.fd             {".sd" is not spelled right}
*.fd not found
>redo              {redo most recent command}
ll *.fd              {last command is printed}
     s               {you enter changes to it}
ll *.sd              {the edited command is shown}
                     {you press Return}

>listredo all
>redo 5        {redo 5th command in stack}
>redo          {redo previous command}
>redo -2       {redo command before previous}
>redo 8/10     {redo 8th through 10th}
>redo -10/     {redo -10 through last}
>redo rm       {redo last "rm" command}
>redo rm temp   {redo last "rm temp"}
>redo @temp   {redo last containing "temp"}
```

Notes

The Redo command can be abbreviated to a comma, as in MPEX.   You cannot use ";" to combine commands on the same line.

Hpmodify Keys - Reference

Directive	**Effect**
i	INSERT. If text follows the i, this text is inserted in the current line starting at the i position.
r	REPLACE. If text follows the r, this text replaces the same number of characters in the current line beginning at the r position.
d	DELETE. Deletes a character from the current line for each d specified in the edit line. Note that "d d" does not specify a range as it does in MPE V but simply deletes one character above each d. Multiple d's may be followed by an INSERT or REPLACE operation.
d>	DELETE. Deletes to the end of the current line from the position specified by d>. May be followed by an INSERT or REPLACE operation.
>	APPEND. If text follows the >, this text is appended to the end of the current line. If a > without text is positioned beyond the end of the current line, then a simple replacement is performed instead.
>d	DELETE. Deletes from the end of the current line, right-to-left. Multiple d's and INSERT and REPLACE strings may be specified after >.
>r	REPLACE. Replaces characters at the end of the command line. The last (rightmost) character of the replacement string is at the end of the line.
c	CHANGE. Changes all occurrences of one string to another in the current line starting at the c. The search string and replace string must be properly delimited. A proper delimiter is a non-alphabetic character (such as ' " or /). The substitution is specified as c*delim search-string delim* [*replace-string* [*delim*]]. Omitting the *replace-string* causes occurrences of *search-string* to be deleted, with no substitution.
u	UNDO. A single u in column one cancels the most recent edit of the current line. Using the Undo command twice in a row cancels all edits for the current line and re-establishes the original, unedited line. If u is placed anywhere other than column one of the current line, then a simple replacement is performed. Undo makes sense only if you have a line on which you have performed some editing that can be "undone."
other	Simple replacement. Any other character (not i, r, d, d>, >, >d, >r, c, or u) will be put into the current line at the position above where it is placed, replacing any existing character. Simple replacement also occurs for the editing characters i, r, c, or > if they are not followed by text; or if > appears at or beyond the current end of line.

Examples

Hpmodify Examples

Edit	**Action**
u	First occurrence undoes the previous edits. The *u* must be in column one.
u	Second occurrence undoes all edits on the current line. The *u* must be in column one.
rxyz	Replaces the current text with xyz starting at the position of *r*.
xyz	Replaces the current text with xyz starting at the position of *x*.
ixyz	Inserts xyz into the current line, starting at the position of the *i*.
ddd	Deletes three characters, one above each *d*.
d xyz	Deletes a single character above the *d*, skips one space, then replaces the current text with xyz starting at the position of *x*.
ddixyz	Deletes two characters, then inserts xyz in the current line starting at the position of the *i*.
d d	Deletes one character above the first *d*, skips two spaces and deletes a second character above the second *d*. It does not delete a range of characters, making it unlike the MPE V version of Redo.
d d>xyz	Deletes a single character above the first *d*, skips two spaces and deletes to the end of the line beginning at the second *d*, and then places xyz at the end of the line.
>xyz	Appends xyz to the end of the current line.
>ddxyz	Deletes the last two characters from the end of the current line and then places xyz at the end of the line.
>rxyz	Replaces the last three characters in the current line with xyz.
>ixyz	Appends xyz to the end of the line. In this case, the *i* command is superfluous, because > accomplishes the same result. Using >xyz would be sufficient.
c/ab/def	Changes all occurrences of ab to def, starting at *c*.
c"ab"	Deletes all occurrences of "ab" starting at *c*.
cxyz	Replaces the current text with cxyz, starting at *c*. Because delimiters have not been specified (as they were in the previous two examples), this is a simple replacement with the four characters.

Persistent Redo

Redo commands can be saved in a permanent file and can therefore be used from another session.   You can use the **Set redo** command to specify a filename to save your redo commands.   Please see the Set Redo command for details.

Reset Command [R]

Resets aspects of the current task.

```
RESET          [ ALL | @ ]
[ command [...] ]
```

(Default: Sort/Key/If/List)

Parms
Examples
Notes

Parameters

More than one command can be Reset at once by entering several *commands* separated by a space or a comma. If no parameters are specified, Suprtool cancels the previous Sort, Key, If, and List commands. The other commands remain unchanged.

If ALL is specified, all of the Input and Output commands are canceled and files are closed. In fact, the only options that are not reset to the initial condition are Define, Item, Open and Set options.

Examples

You began to specify a sort, but then you discovered that you specified the wrong database so you decided to start all over:

```
>open oracle demo reader {open demo}
>select * from sales
>sort account
>reset all               {oops, wrong database}
>open oracle fred reader {now we have correct one}
```

In the next example, you entered an incorrect If command:

```
>if delivered>000401     {wrong field used}
>reset if                {only reset the If command}
>if purchased>000401     {     and specify it again}
```

This time both the If command and the Extract commands are incorrect:

```
>if delivered>000401     {wrong field used}
>extract delivered       {   in both commands}
>reset if,extract        {only reset the If and Extract}
>if purchased>000401     {   commands and start again}
>extract purchased,account
```

Notes

By resetting certain commands, other commands are also reset. For example, resetting the Open command resets almost all other commands, except the Output command. Resetting the Select command resets everything except the Open and Table commands. Resetting the Table command resets everything except the Open and Define commands. Resetting either the Define or Item command resets both Define and Item settings.

Select Command   [SEL]

Specify a select statement for the currently open SQL database.   The select command supports all of the features of the select command of the open SQL database.   Only one select command can be specified at a time.

SELECT   *select-command*

Allbase sorts data:

```
>select * from user.account@emp order by ename
```

Suprtool sorts data:

```
>select * from user.account@emp
>sort ename
```

The *select-command* can contain any expression or clause that is supported by the SQL database.   Some features (e.g., ORDER BY) may be done faster by Suprtool (e.g., the Sort command).

Long commands
Performance

Some Select commands can exceed the 256-character command line limitation. The Select command, however, can be considerably larger if you use the $read feature of the Select command.   This feature is similar to the If command $read feature and is invoked by entering the Select command on a line by itself.

```
>select
-ename,salary,tax_paid
-from scott.employee
-order by ename
-//
>
```

The Suprtool prompt changes from ">" to "-" after entering the Select command by itself on a line.   The entire command gets sent to the Select command parser after terminating with two slash marks (//) or a blank line.

You might realize greater performance gains with the Select command if you specify only the columns that you need in your output file for either tables with many items or when you need only a couple of items from a given table.

The following Select command

```
     >select col1,col2,col3 from user.account@emp
```

may be faster than

```
     >select * from user.account@emp
```

Your mileage may vary.

Set Command   [S]

Enables or disables certain operating options within Suprtool.   These options are not reset by Xeq or Reset commands.

SET	[ALLBASE ROWS]	*number*
	[ARITHMETIC]	has no effect in Suprtool/UX
	[BASECLOSE]	not supported in Suprtool/UX
	[BLOCKSIZE]	has no effect
	[BUFFER]	*size*
	[DATE CUTOFF]	*number*
	[DATE FORCECENTURY]	ON|OFF
	[DATE IFYY2000ERROR]	ON|OFF
	[DATE MAPTOPHDATE8]	ON|OFF
	[DEFER]	has no effect
	[DUMPONERROR]	ON|OFF
	[EOFREAD]	has no effect
	[FILECODE]	not supported
	[FILENAME]	*Edit|Export|Help|Hint*
	[FILENAME]	*Link|Outcount*
	[FIRSTREC]	has no effect
	[HINTS]	has no effect
	[IFCHECK]	ON|OFF
	[IGNORE]	has no effect
	[ITEMABBREVIATEDATE]	ON|OFF
	[INTERACTIVE]	ON|OFF
	[LABELLEDTAPEREWIND]	has no effect
	[LIMITS]	Mpe ON|OFF ReadOnly ON|OFF
	[LIMITS]	Tablesize *size*
	[LIST DATE]	*number*
	[LIST PCL]	0|1|2|3|4|5|6
	[LIST TIME]	*number*
	[LOCK]	has no effect
	[NLS]	*number*
	[OPENMODE]	has no effect
	[ORACLE ROWS]	*number*
	[PATTERN]	NEW|OLD
	[PREFETCH]	not supported
	[PRIVMODE]	has no effect
	[PROGRESS]	Percent *number* Minimun *number*
	[PROMPT]	*character*
	[RECOVER]	has no effect
	[REDO]	*filename*
	[SORTFAST]	has no effect
	[SQUEEZE]	has no effect
	[STATISTICS]	ON|OFF
	[SUBSYSTEM]	has no effect
	[SUSPEND]	has no effect
	[USERLABELS]	has no effect
	[VARSUB]	has no effect
	[WARNINGS]	ON|OFF

<u>Allbase Rows</u>

SET ALLBASE ROWS *number* (Default: 100)

If you open an Allbase database, Suprtool reads more than one row at a time when processing the input source. By default, Suprtool fetches 100 rows at a time. You can vary the number of rows that Suprtool fetches with Set Allbase Rows. The minimum number of rows is one and the maximum number is 990. You must specify Set Allbase Rows before entering the Select command.

SET ARITHMETIC *Classic* | *IEEE* (has no effect in Suprtool/UX)

SET BASECLOSE ON (not supported in Suprtool/UX)

SET BLOCKSIZE *size* | <none> (has no effect in Suprtool/UX)

SET BUFFER *size* (Default: 14,336 or 24,576)

Set Buffer varies the maximum number of words that Suprtool allocates for input and output buffers. This buffer size limits the number of data blocks that Suprtool reads at a time.

Suprtool overrides the maximum buffer size specified, if that is necessary to get the task done. For example, when the block size of the file is greater than buffer maximum specified.

The maximum value is 14,336 words in Suprtool/V and 24,576 words in Suprtool/iX. The default value used by Suprtool is the largest size that fits in the available data structures and still allows Suprtool to get the job done. Using Set Buffer to reduce the size of the buffer reduces the size of the stack that Suprtool uses for copy operations. Suprtool/V always uses a buffer size of 4096 words for sorting. This leaves the maximum amount of room in the stack for the sorting operation.

SET DATE CUTOFF *size* (Default: 10)

Date Cutoff tells Suprtool what century to use when Suprtool generates the date value from the $date and $stddate functions. This setting only affects the date values generated by the $date and $stddate function in the If and Extract commands. This does not affect user data.

Versions of Suprtool without Set Date Cutoff would assume 19 for the century for any user-specified $date with a two-digit year.

Now with Set Date Cutoff *xx*, Suprtool assumes the following: a value of 20 for the century if the two-digit year specified in the $date or $stddate functions is less than the value of Set Date Cutoff; a value of 19 for the century if the two-digit year is greater than or equal to Set Date Cutoff.

The default value of Set Date Cutoff is 10. So the default behavior in $date and $stddate is to treat the two-digit years with values of 00..09 as 2000..2009, and the two-digit years with values of 10..99 as 1910..1999.

```
     Cutoff 10     Cutoff 25

     2000          2000
     ...           ...
     2009          2024
     1910          1925
     1911          1926
     ...           ...
     1999          1999
```

We recommend that you always provide a four-digit year when using $date. However, for reasons of backward compatibility, we introduced Set Date Cutoff. See "Set Date ForceCentury" for more information.

Stddate and Set Date Cutoff

When $stddate has to convert from a date in only a two-digit format, the conversion to the four-digit date will use the value in Set Date Cutoff.

For example,

```
     > get sales-detail
     > set date cutoff 15
     > def new-ship-date,1,4,double
     > item ship-date,date,mmddyy
     > ext order-no / sales-amount
     > ext new-ship-date = $stddate(ship-date)
     > out salesinfo,link
     > xeq
```

In this example, if any ship-date has a year of 14 or less, then the century applied to the new-ship-date field will be 20. Ship-dates with a year of 15 or more will have a century of 19 applied.

SET DATE FORCECENTURY   ON | OFF   (Default: OFF)

Set Date ForceCentury On will not allow a yy date to be entered in the $date function, it will force the user to enter a full ccyy date.

```
   >set date forcecentury on
   >item date-field,date,ccyymmdd
   >if date-field >= $date(00/12/10)

 Error:You must specify the century or Set Date ForceCentury off
```

The default value for Set Date ForceCentury is off.

SET DATE IFYY2000ERROR ON | OFF (Default: ON)

By default Suprtool considers dates with a two-digit century component from the $date and $today functions to be invalid when the dates resolve to be greater than 1999 and the If operation is a relative operation (e.g., greater than or equal to). You can control whether Suprtool considers this condition an error by using the following Set command:

```
>set Date Ifyy2000Error Off
```

The following example shows what is considered to be an error by the If command and how the Set command can turn off the error check:

```
>def a,1,6
>item a,date,yymmdd
>if a >= $today(+2000)
           ^
Error:  Cannot use a date beyond 1999 for this format
>set date ifyy2000error off
>if a >= $date(2000/01/03)
```

We have chosen this condition to be an error by default because when the $date function in the If command resolves a date in a yymmdd format to a value beyond 1999, the result is not always a useful value. For example, a December 10, 2000 date in a yymmdd format would have a value of 001210 and comparisons to this value could be logically incorrect.

SET DATE MAPTOPHDATE8   ON | OFF   (Default: OFF)

This Set command changes any Item command reference to phdate to mean phdate8, assisting conversions to the newer phdate format found in PowerHouse version 8.19 and later.

The Set command in the following example

```
     >set date MapToPhdate8 on
```

changes only the reference to phdate8 in the Item command.   It does not change references that already exist in self-describing files, nor does it change the data.

With this setting enabled, any Item command reference such as

```
     >item mydate,date,phdate
```

actually means phdate8.

SET DEFER ON	(has no effect in Suprtool/UX)

SET DUMPONERROR OFF (Default: ON)

With DUMPONERROR, Suprtool attempts to produce a formatted listing of records that cause a database error. The information printed may include the input record number, the output record number and the data values of the record. Suprtool uses current options of the List command to print the data values. If no List command is specified, Suprtool uses the List defaults.

SET EOFREAD ON (has no effect in Suprtool/UX)

SET FILECODE *number*   (not supported in Suprtool/UX)

SET FILENAME Help | Link | Edit | Hint | Export | Outcount

See the chapter "Installing Suprtool" for a complete description of this Set command.

SET FIRSTREC [0 | 1] (has no effect in Suprtool/UX)

SET HINTS OFF (has no effect in Suprtool/UX)

SET IFCHECK OFF (Default: ON)

With Set Ifcheck On, the If command produces an error if any field used in the If command is not contained entirely within the input file record. For compatibility reasons, users may wish to disable this error checking by turning Set Ifcheck Off.

SET IGNORE ON (has no effect in Suprtool/UX)

SET ITEMABBREVIATEDATE   ON | OFF

(Defaults: ON)

The specification of the Date format within the Item command by default expects the entire keyword.   For example for the date format of Calendar, you would have to specify the entire token of Calendar.   If you Set ItemAbbreviateDate On, you would only have to specify CAL for the Calendar date format.

SET INTERACTIVE ON | OFF (Default: depends)

If you run Suprtool from a session, Set Interactive is On. If you run Suprtool from a batch job or with Stdin or Stdlist re-directed, Set Interactive is Off. When this option is On, Suprtool waits for answers to questions and continues processing even if there are errors. When it is Off, Suprtool aborts on any error, assumes the "correct" answer to any question, and generally acts as if there is not an intelligent being typing in the command. Suprtool chooses the "correct" answer, which allows the task to continue. In most cases, this is the default answer. However, there are cases where Suprtool picks a different answer from the default. For example, an "output filename,erase" command has a default answer of "no," but with Interactive Off, Suprtool uses the answer "yes."

However, if you run Suprtool on a remote session that was created from a batch job, Set Interactive is On even though you are NOT interactive. If you wish to have proper batch error processing, your first command after starting Suprtool should be Set Interactive Off. Set Interactive Off is also useful when automating on-line tasks with usefiles:

```
     suprtool -c"set interactive off;use usefile"
```

SET LABELLEDTAPEREWIND ON | OFF (has no effect in Suprtool/UX)

SET LIMITS [MPE ON|OFF] [READONLY ON|OFF] [Tablesize *size*]

(Defaults : MPE ON, ReadOnly OFF Tablesize 1 Megabyte)

When Set Limits MPE is OFF, you cannot execute **any** HP-UX commands (e.g., >!rm).   This is an irreversible option -- once disabled, it cannot be enabled again by the user.

Tablesize
ReadOnly

Table Size

The Table command allows you to load large tables.   Once these tables reach the size of real memory on your machine, performance starts to degrade. Setting Tablesize restricts the total amount of table space to the specified number of megabytes.

It is not necessary to enter both parameters in order to change one.

```
     >set limits tablesize 2       {HP-UX ability unchanged}
```

Due to internal restrictions in Suprtool, the maximum Tablesize is fifteen megabytes.

Read Only

Suprtool normally allows any user with the proper access capabilities to add records to a database. To prevent users from accidentally updating their database, we provide the following setting within Suprtool:

```
     >set limits ReadOnly On
```

The ReadOnly setting, once turned on, cannot be turned off for the current run of Suprtool. This disables all commands that potentially change data for the specified database.

If Set Limits ReadOnly is on, then the Add command in Suprtool will return an appropriate error message.

You can Set Limits ReadOnly on the command line using the -c option. For example the following command file can be used to restrict who has write access to a given database. In this example only the root user is allowed write access:

```
   if [ $USER = "root" ]
   then
     /opt/robelle/bin/suprtool
   else
     /opt/robelle/bin/suprtool -c'set limits readonly on'
   fi
```

SET LIST

Use Set List to configure default values for the List command.   You can configure the default date, time, and format for LaserJet listings.

Date
Pcl
Time

SET LIST Date *number* (Default: 0)

When you select page headings with the List command by specifying a title, each page includes today's date. By default, this date is formatted as mmm dd, ccyy (e.g., Mar 20, 2000). Use Set List Date to specify a different default date format for future List commands (e.g., Set List Date 2). The valid date formats are as follows:

0	mmm dd, ccyy	Mar 20, 2000 (default)
1	yy/mm/dd	00/03/20
2	mm/dd/yy	03/20/00
3	dd/mm/yy	20/03/00
4	dd mmmyy	20 Mar00

SET LIST PCL 0|1|2|3|4|5|6   (Default: 0)

Use Set List PCL to configure the default format for LaserJet listings.   This option defines the List device as a PCL device and indicates the orientation and font for the report.   Set List PCL affects only the List command; it is ignored by the Output command.   PCL stands for Printer Command Language, which is an HP standard for printers.   The LaserJet is one of the first PCL devices to be released by HP.

By default, Suprtool assumes that your List output device is not PCL-compatible (List PCL 0).

PCL 1. To print the Suprtool List output in Landscape mode (across the wide part of the paper) with the tiny Lineprinter font (16.66 pitch or 8 lines per inch), you should do the following (this setting prints 175 columns per line):

```
     >!#  Maximum of 175 columns with this font
     >set list pcl 1
     >!#  You will need LaserJet with proper font cartridge
     >list device laser123
```

PCL 2. To print the listing in Courier font, Landscape mode, 6 lines per inch, and 100 columns wide, use:

```
>set list pcl 2
>list device laserjet
```

PCL 3. This option selects the "standard" Portrait orientation with the Courier font of the LaserJet (80 columns across by 60 lines). You would use Set List PCL 3 when you insert a Font cartridge that overrides the default font (e.g., 92286F cartridge).

PCL 4. Selects Portrait orientation and Lineprinter font of the L cartridge (and others). This option prints 132 columns across the page by 80 lines.

PCL 5. Prints 80 columns on A4 paper by slightly narrowing the space between columns.

PCL 6. Prints tiny letters in Landscape mode on legal-size paper. This gives you 223 columns per line.

The PCL options, with the exception of PCL 5, were designed and tested with North American letter-size paper. Suprtool can adjust the number of rows and columns for each option to match A4 if you add 2000 to the PCL code. You may also select the ASCII character set (instead of the default Roman-8 character set) by adding 1000 to the PCL code. See the List command for more details.

Here is a complete table of the PCL codes:

			A4 paper		Letter-size		
PCL	L/P	Font	Rows	Columns	Rows	Columns	Notes
1	L	lp	58	188	60	175	
2	L	courier	43	110	45	100	
3	P	courier	64	77	60	80	"standard"
4	P	lp	85	128	80	132	
5	P	courier	64	80	60	80	A4-squeeze
6	L	lp	60	223	60	223	legal-size

L/P mean Landscape or Portrait orientation.

SET LIST Time *number* (Default: 1)

When you select page headings with the List command by specifying a title, each page includes the current time. By default, the time is in 24-hour format (e.g., 23:02). Use Set List Time to specify a different default time format for future List commands (e.g., Set List Time 2). The valid time formats are as follows:

0	none		
1	24-hour	23:02	(default)
2	AM/PM	11:02PM	

SET LOCK 0 | 1 | *number* (has no effect in Suprtool/UX)

SET NLS [*number*] (Default=0)

Use Set NLS with files from MPE systems to specify the language for sorting Character-type fields (see Native Language Support). The *number* corresponds to an NLS language; you cannot use the NLS language name. The common language numbers are:

00	Native-3000	07	French
01	American	08	German
02	Canadian-French	09	Italian
03	Danish	10	Norwegian
04	Dutch	11	Portuguese
05	English	12	Spanish
06	Finnish	13	Swedish

SET OPENMODE *number* (has no effect in Suprtool/UX)

SET ORACLE ROWS *number*   (Default: 100)

If you open an Oracle database, Suprtool reads more than one row at a time when processing the input source.   By default, Suprtool fetches 100 rows at a time.   You can vary the number of rows that Suprtool fetches by using Set Oracle Rows.   The minimum number of rows is 1 and the maximum is 990.   You must specify Set Oracle Rows before entering the Select command.

SET PATTERN   NEW | OLD   (Default: NEW)

Prior to Suprtool for MPE version 3.1, there was no method of checking for the "@", "#", "?", or "~" characters in a pattern.   Version 3.1 introduced a new pattern-matching routine, adding an escape character "&", and two new reserved characters "^" and "!".   Old Suprtool tasks that look for the specific characters &, ^, or ! will not work with the new pattern-matching routine. Users who are concerned about this can add the following command to their /opt/robelle/suprmgr file:

```
     >set pattern old
```

SET PREFETCH *number* (not supported in Suprtool/UX)

SET PRIVMODE OFF (has no effect in Suprtool/UX)

SET PROGRESS Percent *number* Minimum *number*

(Defaults: Percent 5, Minimum 50000)

The Set Progress command is used to turn on or off the progress report feature of Suprtool. The PERCENT value specified tells Suprtool by which percentage increment to report the progress messages of any given input or output phase. The allowed range for set progress is from 0 to 25, the default is every 5 percent. If the PERCENT parameter is not specified, then the next parameter is considered to be the PERCENT value. This is to remain compatible with some earlier versions of Suprtool.

The MINIMUM value is the minimum number of records that an input file must have in order for progress reports to print out. If the MINIMUM value is set to 15000, then the input file must have at least 15000 records or no progress messages will print out for the entire task. This value allows the user to turn off progress messages when reading smaller files. The default value is 50000 records. To always have progress messages, just set the minimum value to 0.

Suprtool does not produce any progress messages under the following conditions:

1) Set Progress is zero.
2) Output is to $stdlist via the Output * or List commands.
3) The number of records from the input source is less than the minimum value.
4) The input source is an SQL database.

Suprtool checks whether or not to print a progress message at the end of each buffer. Consequently, not all progress increments are reported for small files or datasets.

Suprtool reports the phase that it is in: whether input phase, sort phase, output phase or combined input/output phase (no sort).

The content of the progress messages is as follows:

Percentage complete
Phase and the total number of records processed

Delta-Sec(Min)	- the time elapsed from the previous message
Wall-Sec(Min)	- the total elapsed time
CPU-Sec	- the total CPU-Seconds at this point

When using the record selection feature of the Input command Suprtool cannot be absolutely certain of the total number of records. Therefore, the percentage calculation is estimated.

SET PROMPT *character*   (Default: >)

PROMPT tells Suprtool to use a different character for prompting.   Any special character can be used as the prompt character.   For example:

```
$/opt/robelle/bin/suprtool
>set prompt %
%open oracle demo reader
```

SET RECOVER ON (has no effect in Suprtool/UX)

Set Redo **[** *filename* **]**

(Default: none)
(Initially: temporary file)

Commands entered at the Suprtool prompt are saved in something called the redo stack. You can recall commands from the redo stack by using other commands such as Before, Do and Redo. By default, the redo stack is stored in a temporary file and discarded as soon as you exit Suprtool. This temporary stack is not preserved across Suprtool invocations.

The new Set Redo command assigns a permanent file as the redo stack, allowing the stack to become available for future Suprtool invocations. For example, to assign the Myredo file as a persistent redo stack, enter

```
>Set Redo Myredo
```

If the file does not exist, Suprtool creates it. Otherwise, Suprtool uses the existing file. All subsequent commands are written to the persistent redo stack. The setting is valid for the duration of the Suprtool session. As soon as you exit Suprtool, the setting is discarded. Next time you run Suprtool, you will get the temporary stack.

If the file name is not qualified, the redo stack is created in the current working directory. This may be desirable if you want to have separate stacks. If you want to always use the same persistent stacks, you should qualify the name.

The Verify command shows which stack is currently in use. If it shows <temporary>, then Suprtool is using the default stack. Anything else is the name of the file used on the Set Redo command.

Concurrency
Qedit

Concurrency

When Suprtool uses the default temporary stack, it is only accessible to that particular instance of Suprtool.   You can run as many Suprtool instances as you need and each one gets its own redo stack.   With temporary stacks, you will never get into concurrency problems.

If you start using a persistent redo stack, however, you might start running into concurrency problems.   A persistent redo stack can be used only by one Suprtool instance at a time.   If you try to use a persistent redo stack that is already in use, you will get the following message:

```
>Set Redo Myredo
The redo file is already in use.
Unable to open file for REDO stack
```

In this situation, Suprtool continues to use the redo stack active at the time and lets you continue working as normal.

Qedit can also have permanent redo stacks. To prevent products from writing to each other's redo stack, it is advisable to have separate stacks for each product by giving them different file names. For example if you use

```
    set redo myredo
```

you will have a redo stack called Myredo for your Suprtool commands. If you exit Suprtool, then run Qedit and supply the same Set Redo command, your Qedit commands will be written to the same file that was used for your Suprtool commands.

SET SORTFAST ON (has no effect in Suprtool/UX)

SET SQUEEZE [ON | OFF]   (has no effect in Suprtool/UX)

SET STATISTICS ON (Default: OFF)

STATISTICS causes Suprtool to print statistics at the end of each task. This can be useful for determining the effectiveness of Suprtool's If and Sort commands versus similar options on the Select command.

SET SUBSYSTEM [ON | OFF] (has no effect in Suprtool/UX)

SET SUSPEND [ON | OFF] (has no effect in Suprtool/UX)

SET USERLABELS OFF (has no effect in Suprtool/UX)

SET VARSUB [ON | OFF] (has no effect in Suprtool/UX)

SET WARNINGS OFF (Default: ON)

Suprtool normally prints warning messages out to $stdlist. You can turn off these messages when you are running from batch by issuing a Set Warnings off command. If you are simulating batch mode with the Set Interactive Off command, you must do the Set Warnings off after the Set Interactive Off.

The default for this setting is On.

Notes

You cannot set multiple options at once.   Instead, use multiple Set commands. For example, to Set Statistics On and to define a PCL device for the List command, you would specify:

```
     >set statistics on;set list pcl 1
```

Sort Command [SO]

Specifies the next sort key via a table column name, or a field in a self-describing file, or a Defined field. See Key command for sort keys specified by explicit byte position. Up to 20 Sort and Key commands can be specified per extract task. The first key entered is the major sort key.

```
SORT   field [(subscript)] [DESCENDING]
```

(Default: Ascending order)

Field

The *field* specified must be a selected item from the selected table, or a defined field, or a field from a self-describing file.

Subscript

If the field is a compound item (e.g., 2X25), the first sub-item is the default if no *subscript* is specified.   You can sort on any sub-item by specifying the *subscript*.   For example,

```
>sort address(2)          {sorts on 2nd sub-item}
```

Descending vs. Ascending Order

By default, sorts are done in ascending order.   *Descending* specifies that the field is to be sorted in descending order.

Examples

The most common use of the Sort command is to specify a sort field of a database field. You may use the Key command to specify all sorts. We recommend that the Sort command be used wherever possible. If the structure of your database changes, your Suprtool tasks still work if sort fields are specified with the Sort command:

```
     >open oracle demo reader
     >select * from sales      {input from a table}
     >sort account             {primary sort field}
     >sort purchased,desc      {newest transactions first}
     >output dsales            {write the sorted records to}
     >exit                     {    a disc file}
```

<u>Disc File</u>

In the next example we sort a disc file.   We create a field using the Define command. Rather than using the Key command, we use the Sort command to specify the sort field.   If the disc file changes, only the Define command must be changed:

```
>input invent,r 80,nolf {input is from a disc file}
>define a,11,2,int      {"A" is an integer that starts}
>output outfile         {      at the 11th byte of Invent}
>sort a                 {sort the input records by the}
>exit                   {    "A" field}
```

Notes

The Verify command shows all of the current command values and the Reset command cancels them.   If you have not defined any sort fields before the Xeq or Exit command, Suprtool performs a copy only, no sort.

Suprtool uses some temporary files during a sort.   It creates the files in the directory specified by the TMPDIR environment.   The size of the files will be equal to the size of the output file.   If Suprtool runs out of disc space during a sort, you can try to specify another directory for TMPDIR.

Subtotal Option of Duplicate Command

Suprtool can subtotal fields in records that have duplicated sort keys.   See the Total option of the Duplicate command for details.

```
>help duplicate
```

Table Command [TA]

Builds a table of values for testing in the If command. There can be up to ten different tables.

TABLE *tablename, itemname, table-keyword, table-values* [,HOLD]

Tables are used by the $lookup function of the If command. The table keywords are Item, File, and Sorted. The total amount of table space is restricted by Set Limits Tablesize. Use the following scheme to select input records based on many data values:

1. Load a table with the values you are interested in.

2. Use the $lookup function of the If command to search the table.

Item
File
Notes

Adding Individual Values to a Table

To add a value for an item to a table, use:

TABLE *tablename*, *itemname*, ITEM, *value* [,*value*]

When you start entering the values for a table, you must enter all the values for that table before starting another table. Once you switch to another table, the previous table is "closed" and you cannot enter anymore values into it.

Parameters

tablename

Any identifier name up to sixteen characters long. This name can be the same as the name of a Define field or database itemname, but we recommend that you choose a unique name.

itemname

An item from the database specified in the Open command or a Define field. This cannot be a real-type item.

value

A specific *value* that must match the type of the *item*. String values are extended with blanks to the length of the item.

Example
Decimal Places

Examples

Suppose that you wanted to look for several part numbers.   You could use the following If command:

```
     >if part = "12345","67890","39201","92308","14892"
```

You could also load a table with the part numbers:

```
     >table part-table,part,item,"12345"
     >table part-table,part,item,"67890"
     >table part-table,part,item,"39201"
     >table part-table,part,item,"92308"
     >table part-table,part,item,"14892"
```

and use a different If command:

```
     >if $lookup(part-table,part)
```

Sometimes you need to look for all records that do **not** have any of a set of values,

```
     >if part <> "12345","67890","39201","92308","14892"
```

You would use the same Table commands, but a slightly different If command,

```
     >if not $lookup(part-table,part)
```

Values with Decimal Places

If the *itemname* for the table has implied decimal places, the Table command accepts decimal points and scales input values.   For example,

```
>item cost,decimal,2                   {two implied decimal points}
>table cost-table,cost,item,10,10.5,10.75
>if $lookup(cost-table,cost)
>out out3                              {select records for 1000,}
>xeq                                   {   1050, and 1075}
```

Adding Values from a File

You may need to look for hundreds of part numbers. The Table command accepts the table values from a file. The file must contain lookup values in exactly the same format as the itemname which describes the data. Duplicates are eliminated as they are loaded into the table. For a table consisting of values from a file use:

TABLE *tablename*, *itemname*, FILE | SORTED, *filename* [,HOLD] [,DATA(*fieldname1*,*fieldname2...*)]

Parms
Examples

Parameters

itemname

The item determines the data-type and length of the key values in the table. You can only load a table from a self-describing file. Suprtool first checks for the field in the self-describing file.

FILE vs. SORTED

The File option assumes that the file of table values is not sorted. Sorting a large file of values is slow. If the file is already sorted, use the Sorted option: Suprtool checks the records to make sure they are in ascending order.

filename

A valid HP-UX file name; the file must be self-describing.

Hold

By default, the Xeq command resets all tables. Use the Hold option when using the same table in more than one extract task. When Hold is specified, the Xeq command does not reset the table. Hold applies to individual tables, not all tables.

```
     >table part-table,part,file,partin,hold
```

Examples

If all of your part numbers are in the file Partin, you use:

```
     >table part-table,part,file,partin
     >if $lookup(part-table,part)
```

File
Sorted

The following example uses Suprtool to create a file of sales orders for customers in arrears. The orders data is in the database, but the customer information is in a disc file. Suprtool reads the disc file and creates a new self-describing file of customer numbers that are in arrears. This SD file is then used to select the orders for these customers from the orders table in the database. The `account` item occurs in both the disc file and the database. When the Suprtool table is loaded, the `account` field information is obtained from the self-describing arrears file.

```
>input customers,r 60,nolf  {disc file}
>def  account,1,8,display
>def  status,40,2
>if   status="30"           {customers in arrears}
>extract account, status
>out  arrears,link          {self-describing output}
>xeq                        {for the Table command}
>open oracle demo reader
>select * from orders       {sales orders}
>table cust-table,account,file,arrears
>if $lookup(cust-table,account)
>output badorders
>xeq
```

In this next example, low inventory items from the Inventory table are saved in the SD file Invent.   We use this file to load a Suprtool table and select the records from the Product database table.   On the table command, we use the "sorted" table-keyword instead of "file" because the Invent file is already sorted.   We then create a new file Lowprods with all the product information of the low inventory items.

```
>select * from inventory
>if   on_hand_qty<10     {select records}
>sort product_no         {sort by key value}
>out  invent,link        {later, use this file}
>xeq                     {in the Table command}
>sel * from product      {contains product description}
>table product-table,product_no,sorted,invent
>if $lookup(product-table,product_no)
>sort product_no
>output lowprods
>xeq
```

Suprtool can load up to ten tables, either from separate files or the same file.   The following example assumes that the files are self-describing.

```
>input customer
>table cust-table,custno,file,custfile
>table zip-table,zipcode,file,custfile
>if $lookup(cust-table,custno) and $lookup(zip-table,zipcode)
>output newcust,link
>xeq
```

Keep in mind that using multiple tables may be more memory intensive and require more resources.

Notes

The Xeq command clears any tables that are not held. Searching huge tables is CPU-intensive. If you do this on-line, it seriously affects other users. While it is possible to load tables as large as 15 megabytes, we suggest that you make sure that there is enough real memory to hold the table. Set Limits Tablesize restricts the maximum size of tables, so you can limit the total amount of table space to a specified number of megabytes.

Total Command [T]

Totals specified fields in the selected input records.

TOTAL *field* [(*subscript*)] [*decimal-places*]

(Default: *subscript*=first sub-item, *decimal-places*=0)

Parameters

Each totaled *field* must be a database table column, or a field from a self-describing file, or a Defined field. Total specifies that a total value for the field be printed after processing the records. There does not have to be any output file (i.e., it can be $null) for a total to be printed. There can be up to 15 totaled fields.

The *subscript* is valid only for compound items. If no *subscript* is specified, the first field of a compound item is totaled. The *decimal-places* provides a decimal point when the final total is printed. If the Item command specifies the number of implied decimal places, the *decimal-places* parameter is not needed. The values within each field are assumed to be aligned.

$File Options

Writing totals to a file is not supported in Suprtool/UX.

Examples

The first example prints the totals for a single field.

```
>in file1, r 40, nolf
>def amount,1,5
>total amount
>xeq
Totals (TUE, OCT 10, 2000,  4:30 PM):
AMOUNT                             611105+
```

The next example is identical to the previous one, except that we qualify the total with the number of decimal places.

```
>in file1, r 40, nolf
>total amount,2
>xeq
Totals (TUE, OCT 10, 2000,  4:31 PM):
AMOUNT                            6111.05
```

The previous example specified the number of decimal places by using the Total command. The next example is the preferred way to specify decimal places because it qualifies the `balance` field with two decimal places for all Suprtool commands.   This example also totals two fields.

```
>input file1, r 40, nolf
>def   balance,11,4,integer
>def   overdue,21,4,integer
>item  balance,decimal,2
>item  overdue,decimal,2
>total balance
>total overdue
>xeq
Totals (TUE, OCT 10, 2000,  4:32 PM):
BALANCE                         143598.16
OVERDUE                          17399.73
```

Sort Break Totals

Please refer to the Duplicate command on how to generate sort break totals.

Notes

You cannot combine the Total command with the Total option of the Duplicate command.

The Total command prints out a date and time stamp on the title line.   This is for audit purposes.

Update Command   [UP]

The Update command is not available in Suprtool/UX.

Use Command   [U]

Specifies a file of commands to be executed as a group.

```
USE[Q]   filename
```

Database
Quiet
Nested
Notes

Database Date Items

A usefile makes your task easier by allowing common commands to be specified once in an external file.   A common reason for usefiles is to isolate Define and Item commands for a database in one place.   This makes future changes easier and prevents mistakes.   In this example, we isolate all Item commands for dates from our database in a Suprtool usefile.

```
>use store.suprtool
define delivered,deliv_date
define purchased,purch_date
item delivered ,date ,yymmdd
item purchased ,date ,yymmdd
```

Quiet Execution

By default, Suprtool displays the commands in a usefile as they are executed. The quiet option is not used in the examples above so that you could see the actual commands inside each usefile.   Suprtool can execute commands *quietly* using the Useq command:

```
     >useq store.use   {no commands are listed}
```

Nested Usefiles

Usefiles may be nested. In other words, a usefile may use another usefile to a depth of ten files. For example if the contents of the Usedef usefile has a references to Useext, both usefiles would be executed:

```
>in dsales
>use usedef
define delivered,deliv_date
define purchased,purch_date
item delivered ,date ,yymmdd
item purchased ,date ,yymmdd
use useext
ext cust-account
ext deliv-date
ext product-no
ext product-price
ext purch-date
ext sales-qty
ext sales-tax
ext sales-total
>xeq
```

Care must be taken when entering Use commands with a stacked command after a usefile reference. For example, if you enter

```
use usedef;def j,1,6,byte
```

the Define command will not be executed until *after* the Usedef and any other nested Use commands are finished.

Notes

The usefile may be an unnumbered, fixed-length file or a Qedit workfile, but no more than 256 characters per record will be processed.   For compatibility with Qedit, Useq can be abbreviated to UQ.

If a Use file ends with an ampersand, Suprtool will assume that you are continuing the current command on the next line, outside of the use file.

Userpause Command [USER]

The Userpause command prints a prompt message on the screen and waits for the user to press a key.

```
        USERPAUSE   [ "string" ]
```

(Defaults: read without a prompt)

Prints the *string* and waits for any key. Leading spaces in the string are printed.

Examples

Examples

In this example, we have a usefile that displays a message and then waits for the user before starting the task.

```
>q
>q "We will select all transactions over $10,000.  Since"
>q "there are many transactions, this task will take"
>q "some time (usually more than fifteen minutes)."
>q
>userpause "Press any key when you are ready to start."
```

Verify Command   [V]

Displays the specifications that you have entered so far.

```
VERIFY      [ ALL | @ | VERSION ]
            [ command [...] ]
```

(Default: Input, Output, Sort, Numrecs, changed Set values)

Parms
Examples

Parameters

More than one command can be verified at once by entering several *command* names separated by a comma or a space. The format of the Verify output is organized into columns wherever possible.

For Verify All, Suprtool prints all of the information concerning the current invocation of Suprtool, including the value of the Set options and the Suprtool version number.

For Verify Version, Suprtool prints out the version information.

Verify with no parameters prints the current values for Input, Output, and Key or Sort commands. It also prints those Set options which are not currently at their default setting.

Examples

```
>v
>verify
>verify input      {current input file}
>verify if         {selection criteria}
>verify all        {all current options}
>verify version    {version number of Suprtool}
```

Xeq Command   [X]

Terminates entry of commands and begins the extract from the input source.

```
XEQ
```

Notes
Examples

Notes

After the Xeq, Suprtool processes the task and accepts commands to specify another task. Compare this with the Exit command, which stops Suprtool after processing the input. After an Xeq command, all parameters of Suprtool are reset to their initial values, except any database that is left open, the Set options, and the Defined fields even though their calculated offsets are not guaranteed to be correct for the next file processed.

Examples

```
>open oracle demo reader
>select * from customer
>output mcstfile
>xeq                    {copies Customer to Mcstfile}
>select * from orders
>output allorders
>xeq                    {copies Orders to Allorders}
>exit                   {terminate, no task to do}
                        {last Xeq could have been Exit}
```

Calculator Command [=]

Evaluates an expression and prints the result in one of several formats.

=*expression* [,O | D | B | H | A | #   % $]

Any command that begins with an equal sign (=) is treated as an *expression* to be evaluated. An expression consists of numbers and operators, followed by an optional display format.

The operators can be addition (+), subtraction (-), multiplication (*), division (/), or exponentiation (**). The value of the expression is printed immediately on $stdlist.

```
=20+15                  {add two numbers together}
Result=35.0
=20*15                  {multiply the same numbers}
Result=300.0
=20-15                  {subtraction}
Result=5.0
=20/15                  {divide, print precise result}
Result=1.33333333333
=20**15                 {20 raised to the 15th power}
Result=.327680000000E+20
```

Order Of Evaluation
Percentages
Display Format
Input Format
Help In Calculator

Order of Evaluation

Unlike most programming languages, the calculator always evaluates the calculation from left to right.   This is similar to an electronic calculator, where each keystroke is operated on immediately.   You can use parentheses to force the calculator to evaluate the expression in a different order.

```
=14+16+15/3          {compute an average}
Result=15.0
=14+16+(15/3)        {add 14, 16, and the result of 15/3}
Result=35.0
=14+((16+15)/3)      {divide 16+15 by 3, then add to 14}
Result=24.3333333333
```

Percentages

A number in the calculator *expression* may be followed by a percent sign (%). The calculator assumes that you want to qualify the number as a percentage.

```
=125*5%             {what is 5% of 125}
Result=6.25
=125+125*5%         {add 5% of 125 to 125}
Result=12.5
=125+(125*5%)       {oops, we needed to change the order}
Result=131.25       {this looks like the answer we wanted}
```

The last two examples show the importance of the order in which calculator evaluates the expression.   We needed to use parentheses to force calculator to evaluate our *expression* in the correct order.

Display Formats

A calculator *expression* may be followed by a comma and a display letter. The default is decimal (#) and the options are hex ($ or H), octal (% or O), double (D), ASCII (A) and binary (B). With these options, the result is treated as a 32-bit integer.

```
    =10,o               {standard octal format}
    Result=%000012
    =-10,o              {negative number in octal}
    Result=%37777777766
    =100,h              {hexadecimal}
    Result=$0064
```

Double
Ascii
Binary

In Double format, calculator prints the double result as two octal numbers (the way they appear in DEBUG).   The first number represents the high-order 16-bits and the second number represents the low-order 16-bits.

```
=10,d                {treat result as two 16-bit octal words}
Result=%000000 %000012
=1000000000,d        {high-order 16-bits are no-zero}
Result=%035632 %145000
=-10,d               {note negative value, 2's complement}
Result= %177777 %177766
```

In ASCII format, up to four characters are printed in Hex, Decimal, and ASCII display format.

```
=$2020,a
Result=$2020: 32,32 :"  "
=%20161 %72145,a
Result=$2071: 32,113:" q"  $7465:116,101:"te"
```

In binary format, the high-order 16-bits are examined. If these bits are not zero, they are printed as two groups of eight bits. A one (1) means that the bit is on and a zero (0) means that the bit is off. The low-order 16-bits are always printed as two groups of eight bits.

```
=10,b                 {high-order 16-bits suppressed}
Result=%(2)00000000 00001010
=-10,b                {note negative value, 2's complement}
Result=%(2)11111111 11111111 %(2)11111111 11110110
=1000000000,b         {high-order 16-bits are non-zero}
Result=%(2)00111011 10011010 %(2)11001010 00000000
```

Input Format

The calculator supports different input formats for numbers. Octal values are prefixed with a percent sign (%) and hex values with a dollar sign ($). An ASCII string of up to 4 characters is entered in quotes. The result of the last calculation is referred to using #.

```
=%12                    {octal 12 or decimal 10}
Result=10.0
=%12,o                  {octal input and octal display format}
Result=%000012
=$10
Result=16.0
=%177766                {octal number that is really negative}
Result=-10.0
="abcd",h
Result=$61626364
=#,a                    {use result of last calculation}
Result=$6162: 97,98 :"ab"  $6364: 99,100:"cd"
```

Programmers who make use of the MPE DEBUG software are often frustrated with the format that Double Integer numbers are printed. DEBUG prints them as two octal numbers. Calculator accepts two octal numbers as input and prints the result in standard decimal format.

```
=%35632 %145000         {treat as one double integer value}
Result=1000000000.0
=%177777 %177766        {negative double integer value}
Result=-10.0
```

Calculator Help

It may be difficult to remember all of the various options that the calculator offers.   For this reason, you can obtain a short description of the calculator by entering the following:

```
=?        {? gives help}
          {prints a summary of = functions}
```

Errors and Warnings

Suprtool prints two types of messages: errors and warnings.   In both cases Suprtool is letting you know that it has encountered a condition of which you may want to be aware.

This appendix describes both kinds of messages and gives a partial list of warning messages.

Errors

Errors are defined as conditions which immediately prevent Suprtool from continuing, or which allow it to complete a task and then stop, because continuing would likely cause undesirable or erroneous results.

When Suprtool detects a serious error condition such as a syntax error in a command, a file system error, or a sort error, it prints an error message. For example,

```
Error:  Unknown command name, try HELP

Error:  Unable to open >OUTPUT file
```

Finding Errors Automatically

If you have software that scans spool files for error conditions, have it look for "`Error: `".

File System Errors

When a file system error occurs, Suprtool prints the HP-UX error message.

An error during processing terminates the current task (exceptions: bad data with an If command when Set Ignore is On).

Arithmetic Trap Abort

If Suprtool should Abort with Parm=99x, an error has been detected in the Arithmetic Trap Routine.   This should never happen, so please report it to Robelle Solutions Technology Inc.

NUMRECS exceeded; some records not processed.

You specified a Numrecs and have reached it.   This condition is considered an error if the input is from a source other than disc.

Command entered is not a valid Suprtool command.

and

MPE access has been disabled.   See Set Limits command.

Normally, commands that are not valid Suprtool commands are passed off to the operating system.   If access to the O/S has been disabled via the Set Limits command, these commands are no longer passed off.   If the user does not precede the command with a colon, we assume that the invalid command was meant for Suprtool.   If a colon precedes the command, we assume that the command was meant for the operating system.   On HP-UX systems an exclamation mark can be used in place of a colon.

Output-ASCII not allowed with Duplicate None Keys

Not all processing options are allowed in all combinations.   The ASCII option of the Output command, which reformats the output record, does not work with Duplicate None Keys. Dup None Keys assumes that the output record has the same data definitions as the input record.

Warnings

When Suprtool detects an unusual situation that it should bring to your attention, it prints a nonfatal warning message.   For example,

```
     Warning:  No input data specified

     Warning:  DATABASE must be RESTORED if System Crashes
```

The following list explains the most common warnings.

Not all sort fields were extracted.
The sort information will not be written to the output Link file.

This warning occurs when you `>output filename,link` and are sorting by a field, but the field is not included in the list of extracted fields.   Suprlink cannot use the file, but it may be a perfectly valid file for other applications.

NUMRECS exceeded; some records not processed.

You specified a Numrecs and have reached it.

Record selection in effect, percentage calculation is estimated.

You specified an Input with record number selection and the percentage complete is estimated.

Glossary

Certain symbols and terms are common to many Suprtool commands; they are defined here, in alphabetical order.

Within the command definitions, certain concepts appear as variable terms that are meant to be replaced by any valid example of the concept. For example, the syntax for the Use command is "Use *filename*". The *filename* term can be replaced by any valid file name (example: Use ABC).

Batch

Suprtool operates in **session** mode or **batch** mode.   In batch, any error message causes Suprtool to quit.   Warning messages do not cause an abort.   If an error occurs, Suprtool returns a non-zero value as its result.

In batch mode, Suprtool does not prompt for missing information as it does in session mode. For example, if the output file is a duplicate file name, Suprtool automatically answers "yes" to the question asking you to purge the existing file.

Pseudo-Batch Tasks

During a canned on-line task, such as passing usefiles to Suprtool, you can "fool" Suprtool into responding YES to operational questions. For example, if one of the canned tasks requires Suprtool to `output myfile,erase`, then Suprtool asks the question

```
     ERASE all records from this OUTPUT file [no]?
```

You can avoid typing "yes" in response to this question by invoking Suprtool with:

`$suprtool <` *filename*

Calculator

Suprtool, Suprlink and Stexport and Dbedit treat any line that begins with an equal sign ("=") as an expression to be evaluated. To add two numbers together:

```
>=125+512
Result= 637.0
```

An expression consists of numbers and operators. The operators can be addition (+), subtraction (-), multiplication (*), division (/), or exponentiation (**). The value of the expression is printed immediately.

Any number can be followed by a percent sign (%). The calculator assumes that you want to qualify the number as a percentage. For example,

```
>=125*5%
Result= 6.25
```

A complete description of the Suprtool calculator is given after the description of the Xeq command.

Column

See **field**.

Command

The Suprtool command names are:

ADD	add records to an Oracle or Allbase SQL database.
BASE	open a specified IMAGE database.
BEFORE	execute previous command line with changes.
CHAIN	read input via IMAGE search fields.
DEFINE	define fields to sort, select, or extract.
DELETE	delete the selected IMAGE records.
DO	redo one or range of commands without modifying.
DUPLICATE	select or remove duplicate records
EDIT	interactive database editing.
EXIT	execute the current task, then terminate.
EXPORT	send commands to STExport.
EXTRACT	extract data or constants from the input.
FORM	show structure of database or SD files.
GET	read input from an IMAGE dataset.
HELP	explain Suprtool commands.
IF	select a subset from the input records.
INPUT	read input from a file.
ITEM	specify the date format of an item.
KEY	specify an arbitrary sort field.
LINK	send commands to Suprlink.
LIST	list the output records.
LISTREDO	list previous 20 commands (last 1000 available).
NUMRECS	reduce file sizes.
OPEN	open an SQL database.
OUTPUT	specify the output file.
PUT	Add the output records to an IMAGE dataset.
Q	display a message to the screen.
REDO	same as BEFORE command, using HP-style modifiers.
RESET	reset specifications.
SELECT	specify an input source with an SQL select command.
SET	enable/disable options.
SORT	specify a sort field by name.
TABLE	build a table for the If command.
TOTAL	total an input field.
UPDATE	update one or more fields in an IMAGE dataset.
USE	execute Suprtool commands in an external file.
USERPAUSE	display message, wait for user to press return.
VERIFY	display values and current Suprtool status.
XEQ	execute the current task, but do not exit.
:*HP-UX command*	execute a command with your logon shell.
=*expression*	calculator.

Not all commands are supported on every operating system.

Compound Item

See **subscript**.

Control Character

You create a control character by holding down the Control key while you strike another key. "Y" plus Control generates Control-Y. These are normally nonprinting characters, but they may do things to your terminal. For example, Control-G rings the bell. For notes on how to change the HP-UX defaults, see the section on Control Characters and stty in the "Running Suprtool Under HP-UX" chapter.

Suprtool uses control characters for a number of purposes:

In the Before command, control characters specify the edit functions: Control-D for delete, Control-B for before, etc.

Control-Y stops execution of the current Suprtool task. Suprtool prints a status report and asks if you would like to stop the operation.

Control-H causes the cursor to backspace one position in the current line.

Control-X cancels the current input line.

Control-S pauses a listing that is printing too fast for you to read.

Control-Q resumes a listing that you have paused with Control-S.

Database

A database in Suprtool is an IMAGE/3000 database or an SQL database. A database is specified in the Base or Open command. Several commands (e.g., Get, Chain, or Select) do not work until a database has been specified. Some commands only work with IMAGE/3000 databases and other commands only work with SQL databases.

An IMAGE database consists of datasets (files) which in turn consist of fields. An SQL database consists of tables or views, and each table or view consists of columns. In Suprtool, a column name can be used anywhere that a field-name is used. The advantage of using a database is that information about the database is automatically available to Suprtool.

The Form command shows the database structure.

Errors

Errors are messages printed by Suprtool indicating a fatal problem in the task which prevents it from completing.   Error messages are further described in Appendix A.

Field

A field is a portion of a record. When you access an IMAGE dataset, this makes Suprtool aware of the IMAGE fields in the dataset. When you access an SQL database with the Select command, Suprtool is aware of each column name (fields and columns are synonymous in Suprtool). The Define command allows you to define new fields or redefine existing fields to have new sizes or data-types. Use Define to get at bytes of interest within existing fields and to give them an appropriate name. Then you can refer to the defined field in other commands (e.g., Extract, If, etc.). The following commands all contain a field:

```
>if balance>10000         {check that Balance field}
                          {    is greater than a threshold.}

>sort account             {sort selected input records}
                          {    on Account field.}

>extract a,b              {format output record with}
                          {    defined fields A and B.}
```

Filename

A **filename** is any valid filename and is used in Suprtool commands to identify the input source, specify the output destination, or to specify an external file to be accessed in the Table or Use command.   File names may be enclosed in quotes.   The following commands all contain file names:

```
>input xyz                {input records are read from the}
                          {    XYZ file.}

>output *out              {the output file is specified by an MPE}
                          {    file command (:FILE OUT;DEV=LP).}

>use supruse              {read commands from the file SUPRUSE.}

>input "872xyz"           {input records from the 872xyz file.}
```

Repeated Field

See **subscript**.

Special Characters

Certain non-alpha and non-numeric characters like > and : have special meaning within Suprtool.   See the descriptions that follow.   As well, the term "special" designates a class of characters in the If command.

*
Equals
Less-Than
Greater-Than
Less-Than-Greater-Than
Equals-Equals
Greater-Than-Less-Than
At-Sign
#
Question Mark
&
:
!
;
Comma
"
(
)
%
/
\

*** Means $Stdinx / $Stdlist or :File Command**

* in the Input command means to read input from $stdinx (MPE only). * in the Output command means to write the output to $stdlist. * at the front of a file name points back to an MPE :File command.

```
>input *                                 {MPE only}

>output *

>:file t;dev=tape                        {MPE only}
>input *t
```

= Means "Equals" or Calculate

= in the If command means "EQUALS":

```
>if customer = "40832"
```

= in commands means calculate something:

```
=10+25                   {evaluate the expression and}
Result= 35.0             {   print the result.}
```

= in the Input command means that the input file has exactly the same format and fields as the specified IMAGE dataset:

```
>input acctfile=actrans    {Acctfile has the same structure as}
                           {   the Actrans dataset.   MPE only}
```

= in the Output command means to write the sorted input file back into itself.

```
>input myfile; key 1,10
>output=input              {MPE only}
```

< Means "Less Than"

< in the If command means is one field "less than" another field or constant value:

```
>if balance < 10000
```
{balance field is less than 10000?}

By combining < and =, you get "less than or equals":

```
>if balance <= 10000
```
{less than or equal to 10000?}

> Means "Greater Than" or "Enter Command"

> is used for two purposes in Suprtool:

As the Suprtool prompt character (e.g., >Input actrec)

To mean *greater than* in an If command (e.g., if balance>10000). Combining > and =, gives >= for "greater than or equal to".

<> Means "Not Equals"

In the If command, use the two characters <> to mean "not equals":

```
     >if status <> "01"
```

== Means "Matches Pattern"

In the If command, use the two characters == when you want to check a field for a **pattern** of characters. For example, to select records where the customer name contains the word "THOMPSON" somewhere, use:

```
     >if name == "@THOMPSON@"
```

>< Means "Mismatches Pattern"

In the If command, use the two characters >< when you want to select records that fail to match a **pattern** of characters:

```
     >if address >< "@CANADA@"
```

@ Means "Match Anything" in a Pattern

The At-Sign character (@) is used in patterns to indicate that Suprtool should accept anything in that position.   For example:

```
     >if name == "@ROBERT@"
```

The @ matches <null> ("ROBERT" is a valid match); it matches one character ("ROBERTA" is a valid match); it matches multiple characters ("ROBERT M. GREEN" and "The ROBERT E. LEE" are valid matches).

# Means Number (in Patterns) or Prompt Character for the Edit Command

# is used in patterns to match a single numeric character:

```
     >if type=="REC##"  {look for "REC" followed by 2 digits}
```

# is used by the Edit command on MPE to prompt for commands:

```
     >edit                    {MPE only}
     #list d-inventory
```

# is used in the Get and Input commands to read every *n*-th record

? Means Alphanumeric (in Patterns) or Prompt for Database Password

? is used in patterns to match a single alphabetic or numeric character:

```
>if type=="BASE??"{look for "BASE" plus 2 alphanumerics}
```

? is used in the Base and Put commands to force Suprtool to prompt for the database password in either session mode or batch mode:

```
>base store,5,?      {prompt for database password}
Password >           {password not echoed}
```

& Means Escape (in Patterns) or Continue Command Line

& is used in patterns to match one of the special pattern characters. For example, the #-character matches a single numeric character. If you need to look for the #-character itself, you would specify &# in the pattern:

```
     >if type=="REC&#"        {look for "REC" followed by "#"}
```

& is used to continue a command line. You may enter commands on multiple input lines by putting an "&" continuation character at the end of the line:

```
     >if status="20" and &         {continue the If command}
         state="AZ","CA","OR"      {select several states}
```

: Means O/S Commands or Bit Selection

: at the start of a command line indicates an operating system command:

```
>:listf      {MPE command, list files while within Suprtool}
>:ls         {HP-UX command, list files while within Suprtool}
```

: is used in the If command for bit selections:

```
>define bitfield,1,2,logical
>if bitfield.(4:2)=3
```

! Means O/S Commands

! at the start of a command line indicates an operating system command.   This only works on Suprtool/UX.

```
>!du          {HP-UX disc usage command}
>!ls          {HP-UX command, list files while within Suprtool}
```

; Means Multiple Commands

; is used to string several Suprtool commands together on a single line:

```
     >input a;output b;xeq
```
{complete "task" is in one line}

, Means a List

, in Suprtool commands is used to separate parameters:

```
    >base actrec.data,3       {open the database exclusively}
    >key 1,4,double           {specify a double-integer key}
    >if acct=764523,456732,98765
```

Commas are optional in some Suprtool commands (e.g., Output).

, is the abbreviation for the Redo command.

,.   is the abbreviation for the Do command.

,, is the abbreviation for the Listredo command.

" or ' Means String

" or ' are the string delimiters in Suprtool (IF NAME="BOB"). Strings that start with " must end with ".

```
    >if name='BOB'   {' is the string delimiter here}
```

(Means Start Parameter

(is used to specify a subscript (see subscript below) or to select a specific range of input record numbers.   (always comes with).

```
>input actrec.data(10/20)   {choose records 10 through 20}

>total budget(2)            {total second repeated field}
```

) Means End Parameter

) is used to complete a subscript or a selected range of record numbers.   ) always comes with (.

% Means Percentage

In the Numrecs command, use % to indicate the number of output records as a percentage of the input file size.

```
>numrecs 10%
```

/ Means Range of Records

/ in the Input and Get commands mean a range of record numbers.

```
     >input cat.dog.mouse(1000/2000)
```

\ Means Range of Fields

```
>extract account \ rating
```

Strings

Suprtool expects all strings to be surrounded by a pair of single or double quotes (' or "). When Suprtool knows the length of a field, it pads strings with trailing spaces. For example,

```
>define  long,1,125        {125 character field}
>extract long="abcef"      {Suprtool adds 120 spaces}
>if      long="abcde"      {Suprtool checks for the}
>output  tempfile,temp     {    120 trailing spaces }
```

Suprtool accepts the null string. Suprtool pads it with spaces, so this is an easy way to see if a field is blank:

```
>if name = ""              {if name is blank}
```

One problem with any tool that accepts strings is how to include a quote mark inside the string. Suprtool offers two solutions:

1. Use the opposite quote mark (e.g., "don't").

2. Whenever two quote marks appear in a string, they are treated as a single quote (e.g., 'don''t').

Subscript

A subscript is used to specify one-of-many fields in a repeated item.   Within IMAGE it is possible to specify repeated fields.   For example:

```
     costs,         5J2;
```

The item COSTS consists of five double integers.   You select one element of a compound field by specifying a subscript in parentheses (the first element is 1, not 0).   For example, if you wanted to select the input records where the second cost was greater than 10000, you would use:

```
     >if costs(2) > 10000
```

The (2) portion of the command is the subscript.   The default subscript is the first sub-item for Total, Define, Sort, and If, but the entire compound item for Extract.   Table does not allow subscripts -- it always uses (1).   The If command has another syntax, using up to three subscripts, allowing you to refer to subfields without Define (see the If command for details).

Tables

Tables are created with the Table command and they are used for testing in the If and Chain commands.   Tables are used by the $lookup function of the If command.   Use tables when you wish to check a data field for many different test values.   You may also use tables to specify the records to search for with the Chain command.

Table can also mean a table from an SQL database.

Warnings

Warnings are messages produced by Suprtool to let you know about nonfatal conditions which might affect your task.   Some common warning messages and their meanings are described in Appendix A.

Yes or No

When Suprtool asks a question that requires a YES or NO answer, "Y", "OUI", "JA", and "SI" are accepted as "YES", and any other answer is considered "NO".

Suprtool Commands

The following list describes the commands that Suprtool understands at the ">" prompt. Continue command lines with "&" and combine commands on one line with "**;**." You can shorten command names to the substring printed in capitals.

Add
Base
Before
Chain
Define
Delete
Do
Duplicate
Edit
Exit
Export
Extract
Form
Get
Help
If
Input
Item
Key
Link
List
Listredo
Numrecs
Open
Output
Put
Q
Redo
Reset
Select
Set
Sort
Table
Total
Update
Use
Userpause
Verify
Xeq
Functions
Modify
Fields
Data-Types
Selection
Configure
Suprmgr
Access
Patterns

Add

The Add command inserts records into an Oracle table.   Before records can be added, an Oracle database must be open and the Oracle table name must be specified.   The specified table name must be a valid table, not a view.

```
    >add customer
```

BAse

Base is available only in Suprtool for MPE.

Before [**start** [**/stop**]] | **string** | [**ALL** | **@**]
Redo commands with chance to modify. (Default: previous)

Chain

Chain is available only in Suprtool for MPE.

Define **field byteposition sublen** [**type**] [**subcount**]

Define a new data **field** (name <33 characters) by specifying an absolute location and format. First byte of input record is 1, not 0. See **Data-Types**. (Default: type = byte, subcount = 1)

```
    >define city,11,10          {byte type}
    >define TransType,1,2,int   {1 = first}
    >def amt,11,2,int,12        {amt repeats 12 times}
```

Define **field fieldname** [**qualifier**]

Define a new data **field** that is relative to the position of an existing database or Defined field. The **qualifier** equals
[**(subscript)**] [**[offset]**] [**sublen**] [**type**] [**subcount**]
The **(subscript)** specifies one sub-item of a compound item such as 5J2; 1 is the first and default sub-item. The **offset** specifies a byte offset from the existing location of **fieldname**; 1 is the default. The **sublen** and **type** override the existing size and data-type. See **Data-Types**. When you specify a **subcount**, the **sublen** is the byte-length of each subfield.

```
    >define itemcount,status,2,int
    >define costtotal,status[2],4,ieee
    >define middle,name(2)
```

DELete

Delete is available only in Suprtool for MPE.

DO [**start** [**/stop**]] | **string** | [**ALL** | **@**]

Repeat previous commands without modifying them.

```
     >do 5/9    {repeat several commands}
     >do if     {repeat last If command}
```

DUplicate **ONLY** | **NONE** **KEYS** [**num**] | **RECORD**
[**COUNT**] [**TOTAL field ...**]

Include or exclude duplicate records for the output file. There are options to count and total the duplicate output records.

```
    >duplicate none keys       {remove dups}
    >duplicate only keys       {keep only dups}
    >duplicate none keys count {count dups}
    >duplicate none keys total SalesAmt
    >duplicate none keys 1     {first sort level}
```

EDit

Edit is available only in Suprtool for MPE.

Exit [**ABORT** | **SUSPEND** | **XEQ**]

Perform task and return to parent process. Exit Abort abandons task. The **Suspend** option is available only in Suprtool for MPE.

```
    >exit           {default = Xeq}
    >exit abort     {stop without executing}
```

EXPort [**stexport-command**]

You cannot use Suprtool's Export command to invoke STExport/UX, but you can run STExport/UX by itself.

```
    /usr/robelle/bin/stexport
```

EXTract **field** [**(subscript)**]
[**= value | = field2 | = expression**]
[**,..**]

Create output records by stringing together fields from the input record and constant values. Extracts are cumulative. Extract can use dates (see Item command) as well as arithmetic expressions.

```
    >ext custnum,transtable(3)
    >ext CustTotal=15000,CustBal(5)=0
    >ext CustName," ",CustAddr
    >ext SalePrice = SalePrice * 1.10
    >ext Day = date-field mod 100
    >ext ByteField = ^7    {= ^G (bell)}
    >ext c = (6000 - cost)
    >ext yesterdate = $today(-1)
    >ext date-field = $date(*/*/*-1)
    >ext ccyymmdd-i2 = $stddate(date-mmddyy)
    >ext num-days = $days(date1) - $days(date2)
    >ext positive = $abs(num-expression)
    >ext int-field = $truncate(real-expression)
    >ext full-name = $trim(first-name) + $trim(last-name)
    >ext lowercase = $lower(city)
    >ext uppercase = $upper(city)
```

EXTract **field1** [**(subscript1)**]\ **field2** [**(subscript2)**]

Specify an inclusive range of fields to extract.

```
    >ext ProductNo \ SalesQty
    >ext SalesAmt(4) \ SalesAmt(6)
```

Form [**SETS** | **ITEMS** | **PATHS**]
[**dataset** | **data-item** | **filename**]

Show information about self-describing files or SQL databases to $stdlist. Only Form and Form Sdfile are supported in Suprtool/UX. The other options are available only in Suprtool for MPE.

`>form`	{list tables}
`>form sdfile`	{fields in self-describing file}

Get

Get is available only in Suprtool for MPE.

Help [**command-name** | **keyword** [**,section**]]

Provide access to the on-line user manual. Type a **keyword** (e.g., Help Access) or a **command-name**. HQ for brief help.

```
    >help          {default is browse}
    >hq list       {quick summary of List command}
```

IF expression

Select a logical subset of the input source through an **expression** that tests one or more **fields**. See **Record Selection** for details.

```
    >if TransCode="01" and amount>1000
    >if $upper(City)="VANC","SEAT","PORT"
    >if StatWord.(4:1)=1 and addr=="@BC@"
    >if a = alpha
    >if $lookup(part-no-table,partno)
    >if date=$today
    >if TotalPrice <> SalesTotal + Taxes
    >if $null(salestotal)
    >if not $null(salestotal)
    >if $invalid(date-field)
    >if $stddate(dt-mmyydd) < $stddate(dt-aammdd)
    >if date-field mod 100 = 01
    >if InDate<=$date(*/*-6/*){6 months ago}
    >if $days(ship-date) - $days(order-date) > 14
    >if $abs(oldprice - newprice) > 10
    >if $truncate(total / months) > 10
    >if $lower(city) = "niagara falls"
    >if $ltrim(last-name) = "Armstrong"
```

Input **file** [**fileinfo**] [**subset**]

The **file** is the name of an HP-UX file. The **fileinfo** options are Reclen, LF, and NOLF. The Reclen option must be followed by the record length (in bytes) of each data record. **Subset** options select only part of the file.

[**(startrec/endrec)**] {range of record numbers}

[**(#n)**] {selects every *n*th record}

```
>input uxfile,reclen 80,nolf
>input sdfile
```
{requires ".sd" file}

ITem   itemname   DATE | DECIMAL   attribute

Define the date format or implied decimal places for an item. For compound items, all sub-items are assigned the same date type or number of decimal places. The **attribute** is a string for Date and the number of decimal places for Decimal. The Date **attributes** describe the format of the date.

yymmdd	: X6 Z6 J2 K2 P8
ddmmyy	: X6 Z6 J2 K2 P8
mmddyy	: X6 Z6 J2 K2 P8
yyyymmdd	: X8 Z8 J2 K2 P10
ddmmyyyy	: X8 Z8 J2 K2 P10
mmddyyyy	: X8 Z8 J2 K2 P10
ASK	: J1
PHdate	: J1 K1 J2 K2
calendar	: J1 K1
yymm	: X4 Z4 J1 K1
yyymmdd	: J2 P8
Oracle	: X7
ccyymmdd	: X8 Z8 J2 K2 P10
ccyymm	: X6 Z6 J2 K2 P8
yyyymm	: X6 Z6 J2 K2 P8
aamm	: X4
aammdd	: X6
mmddaa	: X6
ddmmaa	: X6
ccyy	: X4 Z4 J1 K1
SRNChronos	: X6
mmyyyy	:X6 Z6 J2 K2 P8
yyddd	:X5 Z5 J2 K2 P8
ccyyddd	:X7 Z7 J2 K2 P10
HPCalendar	:J2 K2
EDSDate	:J2 P8
JulianDay	:J2
PHdate8	: J1 K1 J2 K2

```
>item UnitCost,decimal,2
>item InDate,date,ddmmyy
>extract InDate=$today(-1) {yesterday}
>if InDate<=$date(*/*-6/*) {6 months ago}
```

Key **byteposition bytelen** [**type**] [**DESC**]

Define an arbitrary sort field anywhere within the input record. See **Data-Types**. (Default: type = byte, ascending)

```
    >key 1,10           {first ten bytes}
    >key 21,4,double    {double integer}
    >key 25,2,int,desc  {descending order}
```

LINk [**suprlink-command**]

You cannot use Suprtool's Link command to invoke Suprlink/UX, but you can run Suprlink/UX by itself.

```
     /usr/robelle/bin/suprlink
```

List [**STANDARD**]
[**RECORD**]
[**DUPLEX**]
[**TITLE "string"** [**DATE format**]]
[**HEADING "string"** [**"string"...**]]
[**ONEPERLINE**]
[**LABEL**]
[**PCL format**]
[**NOREC**]
[**NONAME**]
[**NOSKIP**]
[**OCTAL** | **HEX** | **DECIMAL**]
[**CHAR**]
[**LEFTJUSTNUM**]
[**RIGHTJUSTNUM**]
[**TIME format**]
[**LP**]
[**DEVICE name**]

Report selected records to $stdlist or to an LP device. **Standard** produces a columnar report, but the default is either a formatted listing or Octal/Char if the file is unstructured. Use **Title** and **Heading** to customize the listing. **Record** prints on attached printer, and **Duplex** prints two sided. $Null is the default output file for List.

```
    >list standard record {to attached printer}
    >list stand title "Overdue Accounts"
```

LISTREDO [**start** [**/stop**] **|** [**ALL | @**] **string**]
[**;ABS | ;REL | ;UNN**] [**;OUT file**]

Display previous commands; ",," is shortcut. The **Out** option is available only in Suprtool for MPE. (Default: last 20 commands)

```
    >listredo 10/40
    >listredo input
```
{all Input commands}

Numrecs **size** | **percentage%**

Specify size of input, output and Sortscr files as an absolute number of entries or as a percent of input size.

```
    >numrecs 100000
    >numrecs 5%        {5% of the input}
```

OPen **ORACLE username password**
ALLBASE dbename owner

Open an SQL database. Both Allbase and Oracle databases are supported.

```
    >open oracle scott tiger
    >open allbase inventory anne
```

Output **file** [**format**] [**APPEND** | **ERASE** | **TEMP**]

Define the name of the output **file** as one of the following: a new disc file (default), an existing file (**Append** or **Erase** option), or "*" for $stdlist. Suprtool/UX does not support temporary files. If you use List, Put or Total, the output defaults to $null. Output produces the same record format as the input source (adjusted by Extract commands), unless you override with **format** keywords:

Format

Examples

DATA	Default **NUM**		J2 only
KEY	Sort keys only	**NUM,KEY**	J2 & sort keys
PRN	PC file format	**NUM,DATA**	J2 & data record
QUERY	Self-describing	**NUM,QUERY**	"numbers" format
LINK	Better self-describing format		
ASCII	Convert from numeric		
DISPLAY	Convert to display		
NOLF	Do not write out line feeds to end of record		
LF	Write out line feeds to end of record		

Converted ASCII fields have a trailing sign (usually blank for positive values). Size depends on the binary field's format:

E2	12 bytes	E4	23 bytes
I1 J1	06 bytes	I2 J2	11 bytes
I3 J3	16 bytes	I4 J4	20 bytes
K1	05 bytes	K2	10 bytes
Z**n**	*n*+1 bytes	P**n**	*n* bytes

```
>output newfile        {build newfile}
 >output *,ascii        {$stdlist display}
 >output cleanfl,erase {existing file}
 >output trans1,append {append to file}
 >output sdfile,link    {self-describing}
 >output lotus,prn      {ready to download}
```

Put

Put is available only in Suprtool for MPE.

Q [**"string"**]

Print a string on the CRT. (Default: print blank line)

```
    >q "Sorting customer records" {quoted}
```

REDO [**start** [**/stop**]] | **string** | [**ALL** | **@**]

Redo command lines; modify using MPE control codes.

```
    >redo input
```
{repeat previous Input command}

Reset [[**ALL** | **@**] | **command** [**...**]]

Reset aspects of the current task. **Reset All** resets all input and output commands, but not Define and Set options. (Default: Reset Delete, Sort, Key, If, List)

```
    >reset if    {reset If command}
    >reset all
```

SELect statement

Specify an input source with an SQL select statement. Use for Allbase or Suprtool.

Allbase:

```
>select * from user.acct@emp order by name
```

Suprtool:

```
>select * from user.acct@emp
>sort name
```

Set **option-name value**

Enable/disable options. Reset won't affect global Set commands in your /usr/robelle/suprmgr file. Verify Set shows current values. See **Configuration Options**.

```
    >set statistics off
```

SOrt field [**(subscript)**] [**DESCENDING**]

Specify a field or a Defined field as a sort field. For compound items, the first sub-item is the default.

```
    >sort CustNumber     {default ascending sort}
    >sort total desc     {descending order}
    >sort StreetAddress(2)
```

TAble **tablename itemname keyword table-values [Hold]**

Build a table of values for testing in an If command. The **tablename** (up to 16 characters) must be unique. **Itemname** is a database, self-describing or Defined field. For compound items, Table uses the first sub-item. The **keyword** is one of the following: Item, File, or Sorted. These break down to **table-values** as follows: Item, a list of explicit values; File, an unsorted self-describing file with one value per record; Sorted, a sorted self-describing file.

Table **tablename itemname ITEM value** [**value...**]
Table **tablename itemname FILE filename** [**Hold**]
Table **tablename itemname SORTED filename** [**Hold**]

```
>table select,ccode,item,"ABCD","FILM"
>if $lookup(select,ccode)
>table select,ccode,file,sdfile
>if $lookup(select,ccode)
```

Total **field** [**(subscript)**] [**decplaces**]
$file file [**APPEND** | **ERASE** | **TEMP**]
$file $list

Compute sum total of a numeric field value in all selected records and print the result with a specified number of **decplaces** (default is 0 unless specified in Item command). $Null is the default output file for the Total command. The **$file** option is available only in Suprtool for MPE. (Default: subscript = 1)

```
    >total custbal,2
```

UPdate

Update is available only in Suprtool for MPE.

Use[Q] filename

Execute commands from a Text file or a Qedit file.

```
    >use define6.usefile
```

USERpause **"string"**

Print prompt string; wait for any key.

```
    >userpause "Press Return"
```

Verify [**ALL** | **@** | **VERSION**
| **command** [**...**]]

Show current specifications.

```
    >verify input    {print name of input file}
```

Xeq

Perform the specified task, then wait for more commands.

More Suprtool Functions

OS commands `>!rm abc`
`>!chmod 777 myfile`
Calculations `>=312/4` {try "=?" for help}
Sort keys `>10,2,integer`
Comments `>{this is a comment}`

Modify Functions

Printing characters replace; control keys edit. Move with the spacebar. Press the Return key after each function.

^A	Append to end of line (also ^L).
^B	Insert before a column (^^ on console).
^D	Delete (default = end of line).
^G	Restart with original line.
^O	Overwrite characters (default).
^T	Stop current editing mode.
^A,^D	Delete from end of line, then add.

Suprtool Fields

Suprtool recognizes fields in SQL databases and self-describing files. The Define command can define new fields or redefine existing fields into new sizes or data-types so that you can name and use important parts of existing fields. Refer to the Defined field in other commands (e.g., Extract, If, etc.).

```
    >define trans,partnum[7],2
    >if trans="XT"   {last 2 bytes of X8}
```

Data-Types

The data-types for the Define and Key commands:

byte	ASCII characters	**packed**	packed-decimal
integer	two's complement	**packed***	last nibble unused
double	two's complement	**display**	zoned-decimal
real	not supported	**logical**	unsigned integer
long	not supported	**character**	for NLS
ieee	IEEE floating-point		

Definitions

The Suprtool definitions:

Suprtool Definition	**Bytes**	**COBOL Declaration**	**IMAGE Type**
>def a,1,2,integer	2	S9(4) comp	I1
>def a,1,4,double	4	S9(9) comp	I2
>def a,1,8,integer	8	S9(18) comp	I4
>def a,1,2,integer	2	S9(4) comp	J1
>def a,1,4,double	4	S9(9) comp	J2
>def a,1,8,integer	8	S9(18) comp	J4
>def a,1,*n*,byte	n	A(*n*)	U*n*
>def a,1,*n*,byte	n	X(*n*)	X*n*
>def a,1,*n*,display	n	9(*n*)	Z*n*
>def a,1,*n/2*,packed	n/2	S9(*n*-1) comp-3	P*n*
>def a,1,2,logical	2		K1
>def a,1,4,logical	4		K2
>def a,1,4,ieee	4		E2
>def a,1,8,ieee	8		E4

Allbase Data-Type	**Suprtool Data-Type**
Integer	Double
Smallint	Integer
Binary	Not Supported
Char	Byte
Varchar	Byte
Real	IEEE-32
Float	IEEE-64
Decimal	Packed
Numeric	Packed
TID	Not Supported
Date	Byte
Time	Byte
Datetime	Byte
Interval	Byte
Varbinary	Not Supported
Long binary	Not Supported
Long varbinary	Not Supported

The date and time fields are returned as byte-type fields with the following lengths:

Data-Type	**Length**
Date	10
Time	8
Datetime	23
Interval	8

Oracle Data-Type	**Suprtool Data-Type**
Varchar2	Byte
Number	Varies, see **Precision** below*
Long	Not supported
Rowid	Not supported
Date	Oracle Date
Raw	Not supported
Long raw	Not supported
Char	Byte

Mislabel	Not supported

The following table describes the translation for each case:

***Precision**	**Decimal Places**	**Suprtool Data-Type**
None	Any	8-byte IEEE
1-4	Zero	2-byte Integer
5-9	Zero	4-byte Integer
1-9	Non-zero	Packed-decimal
10-27	Any	Packed-decimal
28-38	Any	8-byte IEEE

Record Selection

The If command lets you select a subset of the input source based on **field** values in the entries, and it allows you to combine tests.   Operator precedence is parentheses, NOT, AND, OR, unary minus, multiply and divide, add and subtract.

```
      field = value     {or < > <= >= <>}
      field == "pattern" {see Pattern Matching}
      byte-field = NUMERIC | ALPHA | SPECIAL
      field = value-list   {If A=1,2,3}
      field = field
      (expression)     {parentheses}
      NOT expression   {reverse test}
      expression   AND   expression
      expression   OR     expression
      $invalid(fieldname)
      $stddate(date-field)
      $lookup(tablename,field)
      date-field = $today                {see Item}
      date-field = $date(yy/mm/dd)   {see Item}
      word-field.(bit:count)   {bit extracts}
              {bits go from 0 to 15, left to right}
      byte-field = ^7                      {Control-G}
      $read
      $null(fieldname)
```

Configuration Options

Override Suprtool's default options with Set commands.

+ Supported on Suprtool/UX
- Not supported on Suprtool/UX

Set	Option Name	Sub-Option	Value	Default
+	ALLBASE	ROWS	**number**	100
-	ARITHMETIC		CLASSIC|IEEE	see manual
-	BASECLOSE		ON|OFF	OFF
-	BLOCKSIZE		**size**	<none>
+	BUFFER		**size**	14,336 or 24,576
+	DATE	CUTOFF	**size**	10
+	DATE	FORCECENTURY	ON|OFF	OFF
+	DATE	IFYY2000ERROR	ON|OFF	ON
+	DATE	MAPTOPHDATE8	ON|OFF	OFF
-	DEFER		ON|OFF	OFF
+	DUMPONERROR		ON|OFF	ON
-	EOFREAD		ON|OFF	OFF
-	FILECODE		**number**	0
+	FILENAME	**Edit|Export|Help|Hint|Link|Outcount**		
-	FIRSTREC		**0|1**	see manual
-	HINTS		ON|OFF	ON
+	IFCHECK		ON|OFF	ON
-	IGNORE		ON|OFF	OFF
+	INTERACTIVE		ON|OFF	depends
-	ITEMABBREVIATEDATE		ON|OFF	ON
-	LABELLEDTAPEREWIND		ON|OFF	ON
+	LIMITS	MPE	ON|OFF	ON
+	LIMITS	READONLY	ON|OFF	OFF
+	LIMITS	TABLE	**size**	1
-	LOCK		**0|1|number**	1
+	NLS		**number**	0, Nldatalang
-	OPENMODE		**number**	1
+	ORACLE	ROWS	**number**	100
+	PATTERN		NEW|OLD	NEW
-	PREFETCH (MPE/iX)		**number**	2
-	PRIVMODE		ON|OFF	ON
+	PROGRESS	PERCENT	**number**	5
+	PROGRESS	MINIMUM	**records**	50,000
+	PROMPT		**character**	>
-	RECOVER		ON|OFF	OFF
+	REDO		**filename**	<temporary>
-	SORTFAST (MPE/iX)		ON|OFF	ON
-	SQUEEZE		ON|OFF|<none>	<none>
+	STATISTICS		ON|OFF	OFF
-	SUBSYSTEM		ON|OFF	OFF
-	SUSPEND		ON|OFF	varies
-	USERLABELS		ON|OFF	ON
-	VARSUB (MPE/iX)		ON|OFF	OFF

- WARNINGS	ON|OFF	ON

List

+ LIST TIME **format** 1

+ LIST DATE **0-4** 0

0 = Sep 20, 1994 1 = 94/09/20 2 = 09/20/94

3 = 20/09/94 4 = 20 Sep 94

+ LIST PCL **number** 0

Number (+1,000=ASCII,+2,000=A4)	Letter-size (cols/lines)	A4 paper (cols/lines)
0 default font	80 x 60	
1 Landscape-tiny	175 x 60	188 x 58
2 Landscape-Courier	100 x 45	110 x 43
3 Standard Courier	80 x 60	77 x 64
4 Portrait-tiny	132 x 80	128 x 85
5 Courier A4 "tight"	80 x 60	80 x 64
6 Landscape-legal-tiny	223 x 60	223 x 60

/usr/robelle/suprmgr

This usefile is executed every time Suprtool runs.

```
     set statistics on
     set progress 10 minimum 100000
```

Ways to Run Suprtool

```
/usr/robelle/bin/suprtool [option]
```

Option	**Meaning**
-c	Execute specified command string.
-v	Exit with verify.
-oc	Write count to specified file.

Pattern Matching

The If command can search a byte field for a pattern of characters.

```
   if bytefield == "pattern"     {match}
   if bytefield >< "pattern"     {does not match}
```

@ Zero or more characters (any)	& Escape character
~ Zero or more blanks	! Reserved for future use
? One alphanumeric character	^ Reserved for future use
# One numeric character	

```
    >if item-desc >< "@bolt@"
    >if address=="&#101@"    {look for # sign}
```

How to Contact Robelle

In the United States, in Canada, and in places not listed below, contact us at the following address:

Robelle Solutions Technology Inc.
Suite 201, 15399-102A Ave.
Surrey, B.C. Canada   V3R 7K1

Toll-free:	1.888.robelle
:	(1.888.762.3553)
Phone :	604.582.1700
Fax :	604.582.1799
E-mail :	solutions@robelle.com
E-mail :	support@robelle.com
Web :	www.robelle.com

For our international distributors listing, note that the phone and fax numbers shown are for out-of-country dialing.

Europe
Africa
Asia and Australia
North America

Europe

France, Belgium

ARES
Attention: Renee Belegou
Phone: 33 1 69 86 60 24
Fax: 33 1 69 28 19 18
E-mail: rbelegou@ares.fr
Web: www.ares.fr

Germany

SWS SoftWare Systems GmbH
Attention: Renate Pfund
Phone: 49 7621 689 190
Fax: 41 31 981 32 63
E-mail: info@sws.ch
Web: www.sws.ch

The Netherlands, Belgium

Samco Automation b.v.
Attention: Marius Schild
Phone: 31 13 5215655
Fax: 31 13 5288815
E-mail: marius@samco.nl
Web: www.samco.nl

Nordic Countries

Ole Nord AB
Attention: Ole Nord
Phone: 46 8 623 00 50
Fax: 46 8 35 42 45
E-mail: info@olenordab.se
Web: www.olenordab.se

Switzerland, Austria

SWS SoftWare Systems AG
Attention: Renate Pfund
Phone: 41 31 981 06 66
Fax: 41 31 981 32 63
E-mail: info@sws.ch
Web: www.sws.ch

United Kingdom, Ireland

Robelle Consulting
Attention: Clive Oldfield
Phone: +44 20 7473 2558
Fax: +44 20 7473 2558
E-mail: robelle_oldfield@email.msn.com

Africa

South Africa

Synergy Computing (Pty) Ltd
Attention: Paul Howard

Phone:	27 21 685 7809
Fax:	27 21 685 7927
E-mail:	synergy@synergy.co.za

Asia and Australia

Australia, New Zealand

MRFM Pty. Ltd.

Attention:	Michael Redmond
Phone:	+61 3 9629 8633
Fax:	+61 3 9629 8062
E-mail:	mredmond@mrfm.com.au
Web:	www.mrfm.com.au

Hong Kong

SCS Computer Systems Ltd.

Attention:	Steven Lai
Phone:	852 2609 1338
Fax:	852 2607 3042

Singapore, Malaysia

Singapore Computer Systems Ltd.

Attention:	Toh Tiau Hong
Phone:	65 441 2688
Fax:	65 441 2811
E-mail:	tohth@scs.com.sg
Web:	www.scs.com.sg

North America

Mexico

Infosistemas Financieros SA de CV
Attention: Anita De Urquijo

Phone:	52 5 813 1325
Fax:	52 5 813 3026
E-mail:	adeurquijo@if.com.mx
Web:	www.if.com.mx