



Welcome to STExport

Welcome to STExport for HP-UX Version 4.4 STExport converts fields in a self-describing input file into an output file that can be imported into different applications.

Summary of the STExport commands:

Before	Form	Quote	Verify
Columns	HEAding	REDO	Xeq
DAte	HElp	Reset	Zero
DElimiter	HTML	Set	<i>=expression</i>
DO	Input	Sign	<i>:OS command</i>
Exit	LISTREDO	SPaces	
FLoating	Output	Use	

[Keywords](#)
[Manual](#)
[Notation](#)
[Installation](#)
[Access](#)
[Intro](#)
[Commands](#)
[Example Output](#)
[Limits](#)
[Quick](#)
[Support](#)

Keywords

Everything about STExport is organized under a few keywords:

KEYWORDS	this list of keywords
MANUAL	printing the manual and using it
NOTATION	common notation in STExport commands
ACCESS	how to run STExport; different run options
INTRO	purpose of STExport
COMMANDS	explanation of all STExport commands
FILENAMES	how to adapt built-in file names
INSTALLATION	how to install STExport
EXAMPLE OUTPUT	show the fields in a self-describing file
LIMITS	lists maximum record sizes
QUICK	brief syntax reminders
SUPPPORT	how to contact us for product support

Documentation

Printed copies of the Suprtool user manual can be ordered directly from Robelle or one of its distributors.

The user manual covers the commands of STExport and answers questions that might arise during STExport execution. The user manual is also available to the on-line STExport user through the Help command. For instructions, try:

```
$help help
```

Notation

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

```
Listredo [ start [ / stop ] ] | [ ALL ] | [ string ]
```

1. UPPERCASE LETTERS are required elements in the command, and must be typed exactly as they appear.
Example: ALL
2. *Highlighted lowercase letters*, underlined or italic, are "variables" to be filled in by the user. In the help file, underlining and italics are not available and variables will appear simply in lowercase.
Example: *start*.
3. [] - Brackets enclose optional fields.
Example: [*string*]
4. {} - Braces enclose comments in examples. Braces are allowed for comments in actual STExport commands.
Example: \$Floating fixed 2 {Floating option}
5. | - Up lines separate alternatives from which you will select. Sometimes, the alternatives are shown listed on several lines.
Example: [ALL] | [*string*]
6. In examples, there is an implied carriage return at the end of each line.

Installing STExport

STExport is installed as part of the Suprtool installation process. See the "Installing Suprtool" chapter of the *Suprtool User Manual* for more details on how to install both Suprtool and STExport.

Accessing STExport

To get into STExport, type the following command:

```
/opt/robelle/bin/stexport  
STExport/Copyright Robelle Solutions Technology Inc. 1999  
(Version 4.4)  
$
```

After a short pause, STExport will take over your terminal and print out some identifying information. You will notice that your command prompt has changed "\$", telling you that you have made it into STExport. STExport expects you to type command lines, ending each one with Return.

Xeq

File names

Batch

How to Xeq an STExport Task

Normally, you enter a series of commands. These commands specify the Input file, the Output file, and the formatting options. Finally, you enter an Xeq or an Exit command. This begins the actual STExport task.

If you entered the Exit command, STExport finishes the current task, then returns you to the operating system.

```
$EXIT  
>
```

If you entered the Xeq command, STExport finishes the current task, then prompts you for another task. This continues until you enter the Exit command. If you wish to terminate STExport immediately (perhaps you are confused), enter Exit Abort. This terminates the STExport program immediately, without attempting any task.

Hardcoded File Names and ROBELLE Variable

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

ROBELLE Variable

Normally STExport looks files in the /opt/robelle directory. If you move STExport you must set the ROBELLE variable. For example, if you move STExport to the /users/robelle directory you must set ROBELLE variable in the following manner:

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

Using STExport in Batch

You normally run STExport as an on-line session. You type STExport commands on your terminal and STExport prints responses on your terminal. If you redirect stdin or stdout, STExport assumes that it is in batch. STExport in batch is almost identical to STExport on-line, except for answering questions. When STExport asks a question in batch, no one is there to answer it. Therefore, STExport does not expect an answer from stdin. STExport assumes that you want your batch task to complete, so it always selects the option that will complete the command successfully. This is normally a "YES" answer, as in "yes, purge the file". STExport prints the question on stdout, as well as the answer that it has selected for you.

Introduction to STExport

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

[Input File](#)

[data-types](#)

[Formatting Commands](#)

[Retained Settings](#)

[Performance](#)

Input File

STExport reads one input file and formats each input record into one record in the output file. The Input file must be a self-describing file (use the Output-Link option in Suprtool).

Dates and Decimal Places

Dates and Decimal Places

Use Suprtool's Item command to specify date formats and the number of implied decimal places when you create the self-describing file. STExport uses this information to correctly format the information in the output file. See Appendix A for a complete example of how to use Suprtool's Item command and Output-Link option to create an input file for STExport.

Data-Types

Each STExport formatting command applies to all fields of a specific data-type (i.e., you cannot specify formatting field by field, only by type). For example, all numeric-type fields can be formatted the same.

The main data-types that STExport identifies are

- Byte-Type:** STExport assumes that character information is stored in byte-type fields. By default, all byte-type fields are surrounded by quotes and trailing spaces are removed.
- Numeric-Type:** The numeric data-types are integer, logical, floating-point, packed, and display. STExport converts the internal representation of each data-type into a string of ASCII digits. By default, all numeric-type fields have a leading sign and are variable length. Where appropriate, numeric-type fields are converted with a decimal point.
- Floating-Type:** All commands that affect numeric-type fields also affect floating-type fields. In addition, you can use the Floating command to specify the format and decimal places for floating-type fields (i.e., Classic or IEEE floating-point numbers).
- Date-Type:** If a field has a date format, STExport does extra formatting. By default, dates are formatted into yyymmdd (e.g., 20001125).

Formatting Commands

Use the following table to determine which command applies to which data-type:

Command	Data-Type
Date	date-type
Floating	floating-type
Quote	byte-type
Sign	numeric-type
Spaces	byte-type
Zero	numeric-type

Commands

Many of STExport's commands, such as the formatting commands above, once set will retain their settings between tasks. Several other non-formatting commands will also retain their settings:

Command

Columns
DElimiter
HEAding
HTML

Each command and its options will remain in effect between any STExport task, unless specifically turned off.

For example, if a previous task has had custom Headings set with the Heading and Heading Add options, the Headings will remain in effect for each subsequent task until a new Heading option is entered.

Performance Considerations

On average, STExport is three-to-five times slower than Suprtool. This is the price we pay for having all of STExport's formatting features.

You can make STExport faster by doing the following:

1. Pre-select only the records you need with Suprtool. The fewer records STExport has to process, the faster it runs.
2. Use Suprtool's Extract command to select only the fields that you need to import in your final application. The fewer the number of fields in the input file, the faster STExport can format each record.

STExport Commands

When you run STExport, it prompts for commands on stdlist with a "\$" character and reads command lines from stdin. STExport commands contain a command name followed by one or more parameters, and are patterned after the same commands in Suprtool.

General

Before

Columns

Date

Decimal

Delimiter

Do

Exit

Floating

Form

Heading

Help

HTML

Input

Listredo

Output

Quote

Redo

Reset

Set

Sign

Spaces

Use

Verify

Xeq

Zero

In this chapter, we describe the STExport commands in alphabetical order. Following each command name in brackets is the minimal abbreviation for the command. For example: [I] for Input and [O] for Output.

Abbreviate

Case

Comments

HP-UX Commands

File names

Calculator

Control-Y

Abbreviating

You may shorten the command to the first letter of the command name.

<code>\$v</code>	<code>{verify}</code>
<code>\$x</code>	<code>{xeq}</code>

Uppercase or Lowercase

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

```
$EXIT  
$exit
```

Comments on Command Lines

Comments may appear at the end of any command line, when they are surrounded by braces. Many of the examples in this manual show comments at the end of each command line. You can enter a comment as the only item in a STExport command line. When continuing command lines, the comment can appear before or after the continuation character.

```
  ${ format reals with two decimal places. }  
$input invoices  
$floating fixed 2      {Floating option}  
$output invfile      {produces the file we want}  
$exit
```

HP-UX Commands

STExport accepts HP-UX commands, with or without an exclamation point.

```
$!ls
$ls
```

For commands that are the same in both STExport and HP-UX, STExport only executes the HP-UX command if you type the exclamation point. For example:

```
$set          {you get STExport set command}
$!set        {you get HP-UX set command (ksh)}
```

STExport/UX executes any HP-UX command (e.g., ll), or script file.

File Names

STExport's Input and Output commands accept any valid HP-UX file name. File names are currently limited to 240 characters.

Calculator

Any command line beginning with an equal sign (=) is treated as a calculator expression. This feature can be used to do other calculations without the need of an electronic calculator.

You can obtain a short description of the calculator by entering the following:

```
=?      {? gives help}  
        {prints a summary of = functions}
```

For a detailed description of the calculator and its options, see the Suprtool manual.

Control-Y

You can interrupt a STExport task with the Control-Y key (hold down Control while striking Y). STExport responds by telling you how far it has gotten (IN=, OUT=, etc.), and asking if you wish to stop. Hit the Return key to continue or type `YES` to stop the task.

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.

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; 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          {".fd" 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.
- 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.

Columns Command [C]

Specify whether fields are formatted into variable- or fixed-length columns.

COLUMNS FIXED | NONE

(Default: None)

Most PC software expects imported data to be in variable-length columns. Other database systems prefer data to be aligned in fixed columns. Use the Columns command to specify whether the output file has variable- or fixed-length columns.

Output File

Output File

The Columns command also affects the format of the Output file. If you specify Columns None, the output file will have variable-length records. If you specify Columns Fixed, the output file will have fixed-length records.

Date Command [DA]

Specify a specific date-format for all dates.

```
DATE      NONE | date-format [ "separator" ]  
INVALID  ASTERISKS | NULL | "string"
```

(Default: yyyyymmdd Invalid Asterisks)

Use the Date command to specify an output format for dates. Use the Invalid option to specify how invalid dates should be formatted in the output file. The advantage of the None option is that it formats all dates, whether they are valid or not. If you select a *date-format*, the default Invalid option replaces invalid dates with asterisks "*".

STExport must know which fields are dates and the format of each date. Use Suprtool's Item command and Output,Link option to specify the date information.

Date Format

Invalid Dates

Example

Date Format

The *date-format* can be one of:

ccyymmdd
yyyymmdd
ddmmyyyy
mmdyyy
yymmdd
ddmmyy
mmdyy
aammdd

STExport converts each date field from its internal date format into the format that you specify.

Separator
Oracle

Separator Character

By default, STExport formats all dates without a separator between the day, month, and year. Specify your own separator by enclosing it inside quotes after you specify the date format. The separator must be one character long. For example, to specify dates in ddmmyyyy format with a slash separator, use

```
$date ddmmyyyy "/"
```

To specify dates in yymmdd format with a dash separator, use

```
$date yymmdd "-"
```

Oracle Dates

Oracle dates contain both the date and the time. STExport formats the date, but not the time. If you specify Date None, Oracle dates will be treated as byte-type fields. Since Oracle dates actually contain binary data, the output is often unusable by other applications, unless you specify a specific *date-format*.

Invalid Dates

By default, all invalid dates are formatted as asterisks. STExport treats any date that does not have a valid century, year, month, or combination (e.g., February 29, 2000) as invalid. You can specify how you want STExport to format invalid dates by using the Invalid option of the Date command.

If you specify,

```
$date invalid null
```

STExport will produce a zero-length field if you specify Column Variable and spaces if you specify Column Fixed. If you want to specify an explicit string for all invalid dates, do so after the Invalid option. For example,

```
$date invalid "%%%"
```

will cause STExport to produce a string of five percent signs for any invalid date.

Example

First, use Suprtool to create the input file with the appropriate date attributes:

```
>get      d-sales
>item    deliv-date, date, mmddyyyy
>item    purch-date, date, mmddyyyy
>output  dsales, link
>xexq
```

Then use STExport to read the dsales file. Specify Date ddmmyyyy "-" which causes all valid dates to be formatted in day-month-year format with a dash as the separator:

```
$input  dsales
$date   ddmmyyyy "-"
$output dexport
$xexq
```

Decimal Command [DEC]

Specify the format for the decimal place in numeric fields.

DECIMAL PERIOD | COMMA

(Default: Period)

The Decimal command specifies what separator will be used to indicate the decimal place in numeric fields. In North America, the custom is to indicate the decimal place in numbers with a period (.). Outside North America, the custom is to indicate the decimal place with a comma (,). If the decimal place indicator is incorrect, it is harder to import files into other applications.

The Decimal command does not apply to floating-point fields.

Delimiter Command [DE]

Specify a delimiter, if any, that appears between each field in the output record.

DELIMITER NONE | COMMA | TAB | SPACE | "*string*"

(Default: Comma)

Use Delimiter Comma to create an output file in "comma-delimited" format (this is common for PC database applications). Use Delimiter Tab to tell STExport to insert the tab character between fields, instead of a comma.

If you have selected Columns Fixed, you will likely want to remove the delimiter by specifying Delimiter None. If you want some white space between fixed-length columns, specify Delimiter Space instead.

String Parameter

String Parameter

You can put anything inside quote characters to specify your own Delimiter. For example, Delimiter " , " would insert a space, a comma, and another space between each field in the output record. You can use either single- or double-quote characters to specify the delimiter (e.g., Delimiter " " and Delimiter ' ' are the same). The maximum length of the delimiter string is three characters.

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 one as entered; 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

Persistence

Examples

<code>\$!listredo</code>	
<code>\$do</code>	<code>{do previous command again}</code>
<code>\$do 39</code>	<code>{do command line 39 again}</code>
<code>\$do 5/8</code>	<code>{do command lines 5 to 8 again}</code>
<code>\$do input</code>	<code>{do most recent Input command}</code>
<code>\$do -2</code>	<code>{do command before previous}</code>
<code>\$do -7/-5</code>	<code>{do by relative line number}</code>
<code>\$do 5/</code>	<code>{do command lines 5 to "last"}</code>

Notes

The Do command cannot be abbreviated.

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.

Exit Command [E]

Exit STExport in one of three ways. By default, perform the current task, if any, then leave STExport. Users are often frustrated when they exit STExport after specifying part of a task and STExport starts processing the task. To avoid this situation, use the Abort or Suspend options to exit STExport conveniently without executing the current task.

EXIT [ABORT | SUSPEND | XEQ]

Typing Exit with no parameters means Exit Xeq. STExport recognizes special command names which specify both the Exit command and an exit option (e.g., ES means Exit Suspend).

Abort

Suspend

Xeq

Exit Abort [EA]

Cancels the current operation and terminates STExport. The Exit command without parameters always attempts to perform the task currently specified, while Exit Abort cancels the task and terminates immediately. Should STExport be executed as a son process, Exit only suspends STExport, while Exit Abort actually terminates the process.

Examples

Examples

```
$:comment.  You began to specify an input file, stopped for  
$:comment.  coffee, and decided to cancel the task  
$:comment.  upon your return.  
$input invoices  
... coffee break ...  
$exit abort          {cancel the task and terminate}  
End Of Program
```


Exit Suspend [ES]

This feature is not currently available in STExport/UX.

Exit Xeq [EX]

To perform the current task, you can either use Xeq (which leaves you inside STExport, ready to define another task) or Exit Xeq (which leaves STExport when done with the task). Exit Xeq is the default option (i.e., specifying exit starts execution of the current task).

Examples

Examples

```
/opt/robelle/bin/stexport  
$exit {no input was specified}
```

```
/opt/robelle/bin/stexport  
$input invoices  
$floating fixed 2  
$output invdata  
$exit {format and stop}
```

Floating Command [FL]

Specify the format and the number of decimal-places for floating-point fields.

FLOATING DEFAULT |
FIXED *decimal-places* |
SCIENTIFIC *decimal-places*

(Default: Default)

By default, STExport formats floating-point fields into either a fixed number or into scientific notation. Which notation STExport chooses, depends on the value of each field in each input record. You can force STExport to choose either scientific or fixed notation and the number of decimal places for all floating-point numbers. You cannot specify these options for a specific field or make them different for 32-bit versus 64-bit floating-point numbers.

Fixed

Scientific

Notes

Fixed Format

Use Floating Fixed to force all floating-point numbers to appear in a fixed format. You specify the maximum number of digits to the right of the decimal point. If you specify Floating Fixed, STExport does not remove trailing zeros from the formatted numbers. If you specify Columns Fixed, all floating-point values will be aligned along the decimal point.

Scientific Format

Use Floating Scientific to force all floating-point numbers to appear in scientific notation. You must specify the number of digits to the right of the decimal point. The Scientific option formats the number with all significant digits to the right of the decimal-point followed by the exponent (e.g., "0.47832E-10").

Notes

Both the Fixed and Scientific options attempt to round the number to the specified number of decimal-places within the maximum width for each floating-point data-type. If STExport cannot format a floating-point field in the specified number of decimal-places, the number appears as asterisks "*****".

Form Command [F]

Display the fields in a self-describing file.

FORM [*filename*]

If no file name is specified, the fields in the input file are displayed. The display shows the field type and field length in IMAGE notation. An I1-field is a single integer. Packed-fields show the number of nibbles (subtract one to obtain the number of digits). Byte and zoned-decimal fields show the byte length.

When showing the form of a self-describing file, STExport shows the byte offset of each field after the subcount, type, and sublength. The first field always appears at offset one.

There are two types of self-describing files. One type is produced with Suprtool's Query output option. You produce the other type with the Link output option. The Form command shows the internal self-describing version number, enabling you to tell the difference.

Query

Link

A.00.00 - Query Output Option

Compound fields have a question mark for the type, and the length is the number of bytes in the field. Sort information about the file is missing. Here is an example form listing:

Example

\$form custfile

File: custfile (SD Version A.00.00) Has linefeeds

Entry: Offset

CHARACTER	X5	1	{length is five bytes}
ZONED	Z5	6	{room for five digits}
INTEGER	I1	11	{single integer}
DOUBLE	I2	13	{double integer}
PACKED	P6	17	{room for five digits}
QUAD	I4	20	{eight-byte integer}
REPEATINT	?6	28	{compound field}
LOGICAL	K1	34	{single logical}
DBLLOG	K2	36	{double logical}

Entry Length: 44 Blocking: 1

B.00.00 - Link Output Option

These self-describing files contain information about how the file was sorted. Compound fields are handled correctly, so the Form command shows compound fields just as you would see them in IMAGE. The Item command in Suprtool identifies the date format or the number of decimal places of an item. The Link output option saves the date and decimal attributes as part of the field description:

```
$form custfile
```

```
File: datafile      (SD Version B.00.00)  Has Linefeeds
```

```
  Entry:           Offset
  CHARACTER        X5    1  <<Sort #1 >>
  REPEATINT        3I1   6  {compound field}
  DATE             J2   12  <<YYYYMMDD>>
  DOLLAR           P6   16  << .2 >>
Entry Length: 16  Blocking: 1
```

Heading Command [HEA]

Specify a heading, if any, that appears as the first record of the output file.

HEADING NONE | FIELDNAMES | *string* | ADD *string* | COLUMN *string*

(Default: None)

When importing data into other applications the first line of the import file is often treated as field names or headings. Use the Heading command to specify what STExport should write as the first line of the output file.

Field names

User Specified

Column Headings

Notes

Field Names

If you specify Heading Fieldnames, STExport creates a default heading. This heading is constructed by using the field name of each field in the input file. The Fieldname option uses the formatting options that apply to byte-type fields to determine the final format (e.g., the Quote command).

STExport produces multiple field names for compound fields. For compound fields, the repeat count is used to determine the number of field names. The repeat count is appended to the field name, starting with one, until all the field names have been generated.

User Specified Heading

You can specify your own heading line by doing:

```
>heading "your heading"
```

Because the maximum length of an STExport input line is 256-characters, you may not be able to specify a long heading with a single Heading command. Use Heading Add to add additional strings to your heading:

```
Heading      "Account      "          {Note no Add in first string}
Heading Add  "First Name  "
Heading Add  "Last Name  "
Heading Add  "City      "
Heading Add  "State     "
```

If you specify your own heading, STExport does not attempt to apply any formatting options. If you need each field in the heading line to be surrounded by quotes and separated by commas, you have to supply these yourself. For example,

```
Heading      '"Account      "'          {Note no Add in first string}
Heading Add  ', '
Heading Add  '"First Name"'
Heading Add  ', '
Heading Add  '"Last Name"'
Heading Add  ', '
Heading Add  '"City"'
Heading Add  ', '
Heading Add  '"State"'
```

Column Headings

It is difficult to get headings right when you have to specify all the quotes and delimiters with the Heading Add option. Instead, use Heading Column to specify individual column headings without having to type formatting information. STExport then uses the current quote and delimiter settings in the heading.

For example, if you specify:

```
Heading Column 'Account '  
Heading Column 'First Name'  
Heading Column 'Last Name'  
Heading Column 'City'  
Heading Column 'State'
```

and Quote Double and Delimiter Comma are in effect, then the heading STExport produces will be:

```
"Account      ", "First Name", "Last Name", "City", "State"
```

Notes

You cannot combine the Add and Column options. You must specify one or the other. If you start with Heading Add and then later specify Heading Column, STExport erases the heading you created with Heading Add and starts over with the first column that you specify with Heading Column. Similarly, if you start with Heading Column, a Heading *string* or Heading Add will start over with a new heading.

Help Command [H]

Show what commands and options are available in STExport.

HELP [*command* | *keyword* [,*section*]]

(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 specific STExport command. If you know the structure of the help file, you can specify one of the keywords under the command name.

```
$help sign           {help on the Sign command}  
$help sign, trailing {trailing section of the Sign 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 STExport and then a list of keywords. You can specify any of these keywords on the Help command. You can also specify a subkeyword.

```
$help intro,input      {input section of Introduction}
```

Quick Help - HQ

HQ asks STExport to look under the keyword Quick in the help file. Quick contains the text from the STExport Quick Reference Guide, offering the experienced user a quick review of the syntax of any command.

`$hq input`

`{quick description of Input}`

Notes

If no parameters are specified, Help allows you to browse through the help file, /opt/robelle/help/stexport. The Help command uses the Qhelp subsystem from the QLIB. 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. Press F8 to exit the Qhelp subsystem and return to STExport.

HTML Command [HT]

Use HTML to produce Web pages for either Internet or Intranet applications.

```
HTML          NONE | PREFORMATTED | TABLE |  
TITLE "string" |  
HEADING "string"
```

(Default: None)

Web applications expect data in a special format called the Hypertext Markup Language (HTML). Use the HTML option to request that STExport format the input file into HTML format.

[Example](#)

[Maximum Size](#)

[Preformatted](#)

[Table](#)

[Title](#)

[Heading](#)

[Column Headings](#)

[Roman-8](#)

[Notes](#)

[Dynamic Web Pages](#)

Example

```
$html table title "Product Listing"
```

Maximum Size of HTML Files

Web browsers often cannot process large documents. The maximum size depends on the browser, the version of that browser, the operating system it's working on, and how much physical memory is present on the client machine. We suggest that you limit your Web pages to less than 1,000 lines and restrict the number of columns, unless you are certain that your users can handle larger files. This advice reflects not a limitation of STExport, but a limitation of how Web browsers work.

Preformatted Format

To preserve the columns and spacing of each output line, use the HTML Preformatted option. This option puts an HTML `<pre>` tag around all the data in the input file. Most Web browsers will display preformatted text in a fixed-width font such as Courier. Therefore, if you specify HTML Preformatted, you should also select Columns Fixed.

Table Format

Use HTML Table to create output in HTML table format. STExport creates tables with a border between each column and row. Tables make it easier to read tabular information, but some older browsers do not support tables.

If you specify HTML Table, all byte-type fields are left-justified and all other fields are right-justified. If you use Heading Column or Fieldnames, the column headings are specified with HTML table heading tags. Most browsers highlight the column headings, such as bold text centered over the column.

Title

All HTML documents must have a title. By default, STExport uses the title "This is the Title". You should specify your own title using the Title option.

Heading

The heading appears before the column headings and data from your input file. By default, there is no heading. Use the Heading option to specify your own heading.

Column Headings

If you specify HTML Table, use the Heading command to specify column headings for HTML output. The Heading Fieldnames option will produce acceptable column headings, but it is better to use Heading Column to specify a string for each of the fields in your input file.

Roman-8 Characters

HP e3000 and HP 9000 computers use the Roman-8 character set. The characters in the Roman-8 set are similar to, but not identical with, the ISO-8859-1 character set. Web pages must use the ISO-8859-1 character set.

When formatting byte-type fields, STExport attempts to convert any Roman-8 input character into the corresponding ISO-8859-1 character. Those characters that cannot be converted are dropped from the output. The following characters cannot be converted:

'	169	grave mark
^	170	circumflex
	172	tilde
f	190	function symbol
ß	222	beta symbol
R	235	capital-S, Icelandic
s	236	small-S, Icelandic
Y	238	capital-Y, umlaut

Notes

If you specify HTML Table, STExport sets:

Quotes None
Delimiters None

If you specify HTML Preformatted, STExport sets:

Quotes None
Delimiters Space
Columns None

In either case, any changes cause STExport to print a warning to let you know that these options have changed. If you do want quotes around byte-type fields or delimiters between fields, specify these options after selecting the HTML option.

Dynamic Web Pages

The Common Gateway Interface (CGI) feature of your Web server allows you to execute custom programs or scripts, and to dynamically generate, then display Web pages. These custom programs and scripts can be written in almost any programming language. Perl is probably the most commonly used language, but you could use C or C++, Applescript on Macintosh, or Visual Basic.

Web Server

First, your Web server software must be configured to allow CGI scripts. On the NCSA server, you need to change the srm.conf file to include

```
ScriptAlias /cgi-bin/ /usr/local/httpd/server/cgi-bin
```

This indicates to the server where the CGI scripts are located. If a user enters "cgi-bin" in the Uniform Resource Locator (URL), the Web server executes the program specified out of the directory specified, e.g.,

```
http://www.mycompany.com/cgi-bin/myscript.pl
```

will actually execute

```
/usr/local/httpd/server/cgi-bin/myscript.pl
```

On the CERN Web server, you need to enter the following in the httpd.conf configuration file:

```
Exec /cgi-bin/* /usr/local/httpd/server/cgi-bin
```

Of course, the alias name and directory can be anything you want them to be.

A script can be coded to perform the same task every time it is run. There are different ways you can pass parameters to a script so that it can do different things. We will try to keep it simple and focus our attention on a specialized program.

Our application will display a list of all sales records created yesterday. The information is stored in an Allbase database. The table is called demo.sales and has the following columns:

Column Name:	Allbase Type:	Nulls:	Suprtool Type:
CUSTOMERNUM	Decimal (8)	N	Packed
DELIVERYDATE	Decimal (8)	Y	Packed
PRODUCTNUM	Decimal (8)	N	Packed
PRICE	Decimal (8)	Y	Packed
PURCHASEDATE	Decimal (8)	N	Packed
SALESQTY	Decimal (4)	Y	Packed
SALESTAX	Decimal (8)	Y	Packed
SALESTOTAL	Decimal (8)	Y	Packed

DeliveryDate and PurchaseDate are in yyymmdd format. Price, SalesTax and SalesTotal have two implied decimals.

Before you go any further, you will have to decide where the new HTML file will reside. When someone connects to your Web site, the server software spawns child processes using a userid and groupid defined in the srm.conf (NCSA) or httpd.conf (CERN) configuration file. This user must have read and write access to the directory specified in STEXport's Output command. This directory must also be accessible by the server software as defined in the configuration file. For security reasons, it is not recommended that you create these files in the server's root directory.

You can configure a default html directory (for example, userdir entry in CERN) for individual users. When the URL contains a `~user` construct, the server knows it has to look for the default directory under the user's home directory. Let us assume the default is `public_html`.

You could create a `public_html` directory under the home directory for the userid specified in the configuration file.

Another option is to create a new user on your system with the same directory structure, whose sole purpose would be to hold these dynamic HTML files. For example, create a user called `htmluser` whose home directory would be `/users/htmluser`. Create a `/users/htmluser/public_html` directory. Directory permissions should be read/write for user, group and others. The URL to read these would then be:

```
http://www.mycompany.com/~htmluser/filename.html
```

Shell Script

Suprtool and STEXport will be executed from a Korn shell script. Let us call it `sx_orders.ksh`. It can reside in any directory that has appropriate permissions to execute. Do not forget to assign eXecute permission to the file (`chmod +x sx_orders.ksh`). You can run this script manually from the `$` prompt to see if there are any problems.

```
#!/bin/ksh
#
# Make sure the PATH variable includes /opt/robelle/bin
# where Suprtool and STEXport usually reside.
#
PATH=$PATH:/opt/robelle/bin

#
# Create a temporary logfile
# and get a file name for the temporary extract file
#
temp_sxlog=`mktemp`
touch $temp_sxlog
temp_sxdata=`mktemp`

#
# Run Suprtool and extract the data.
# - Read commands using "here-is"
# - write messages to temporary logfile
#
suprtool <<!EOD >> $temp_sxlog
Open Allbase /users/orders/db/OrdersDBE clerk
Select * From clerk.sales
Item purchasedate, date, yyyyymmdd
Item deliverydate, date, yyyyymmdd
```

```

Item price, decimal, 2
Item salestax, decimal, 2
Item salestotal, decimal, 2
If purchasedate = \$today(-1)
Sort customernum
Output $temp_sxdata,Link
Exit
!EOD

#
# Check return code in case Suprtool had a problem
# If so, send the "failed" return string to the CGI script
# including the name of the logfile
#
if [ $? -eq 1 ]
then
    echo failed=$temp_sxlog
    exit 1
else

#
# Suprtool task worked correctly.
# Can go on with STExport.
#
# Must go to the target directory
# because of file name limit in STExport
# and get a file name to store HTML output
#
    cd /users/htmluser/public_html > /dev/null
    temp_stexpdata=`mktemp -d .`

#
# Execute STExport to create the html file
#
    stexport <<!EOD >> $temp_sxlog
    Input $temp_sxdata
    HTML Table Title "Orders Created Yesterday" \
        Heading "Orders Created Yesterday (sorted by customer)"
    Heading Fieldnames
    Date YYYYMMDD "/"
    Output $temp_stexpdata.html
    Exit
!EOD

#
# Check return code in case STExport had a problem.
# If so, send the "failed" return string to the CGI script
# including the name of the logfile
#
    if [ $? -eq 1 ]
    then
        echo failed=$temp_sxlog
        exit 1
    else

#
# Everything worked fine.

```

```

# Send the "OK" return string to the CGI script
# including the name of the HTML file name.
#
    echo OK=$temp_stexpdata.html
    rm $temp_sxlog      > /dev/null
    rm $temp_sxdata    > /dev/null
    exit 0
fi
fi

```

Perl Script

With such a simple example, we could have written the shell script a bit differently and then used it as a CGI script. However, shell scripts do not have enough features to easily handle parameters or complex forms.

Instead, it is typical to use another language as an intermediate step. One of the most popular CGI scripting languages is Perl. Perl has a full set of string handling functions and can be combined with graphics libraries available from various sources.

Let us call this one `sx_orders.pl`. It must reside in the directory defined in the server configuration file.

```

#!/usr/local/bin/perl

#
# Execute shell script and capture returned string
#
$return_string = `/users/orders/scripts/sx_orders.ksh`;

#
# Return string can be OK=html file name
#           or           failed=log file name
# Split function separates return_string's 2 components
#
($status_word, $html_file) = split (/=/, $return_string);

if ($status_word eq "failed") {
#
# If the script failed, print an HTML error page
# including the logfile
#
    error_page
}

else {
#
# If the script is successful, display the HTML file
# created by STEXport.
# This URL automatically looks for the configured directory
# under htmluser's home directory.
#
print "Location: http://www.hp.com/~htmluser/$html_file", "\n\n"
}

```

```

# Format an error message for the user
sub error_page {
    print "Content-type: text/html\n\n";
    print "<HTML>\n";
    print "<HEAD>\n";
    print "<TITLE>Comment Form Error</TITLE>\n";
    print "</HEAD>\n";
    print "<BODY>\n";
    print "<H1>Comment Form Error</H1>\n";
    print "<HR>\n";
    print "<P>\n";
    print "Form input was not processed. Please mail your ";
    print "remarks to <b>$webmaster</b>";
    print "<P>\n";
    print "<b>Content of the Suprtool/STExport logfile</b>";
    print "<pre>";
    print `cat $html_file`, "\n";
    print "</pre>", "\n";
    print "</BODY>\n";
    print "</HTML>\n";
}

```

CGI Script

The CGI script can be invoked explicitly when you type the URL in your browser:

```
http://www.mycompany.com/cgi-bin/sx_orders.pl
```

The script is executed either from an input form,

```
<FORM METHOD="POST" ACTION="/cgi-bin/sx_orders.pl">
```

or invoked through a link in a regular Web page

```
Just click <a href="/cgi-bin/sx_orders.pl">run script</a>
```

If you enter the URL, choose the "submit" button on a form, or click on a link, the server will start execution of the Perl script. In turn, Perl will run the shell script. Suprtool and STExport will finally be executed. A status code and a file name will be returned to Perl. It will print either an error page or the extracted information, depending on the status code.

There is a lot more you can do with CGI scripting and Suprtool. You could pass parameters to extract information out of different tables in Oracle and Allbase, or specify the sort sequence and the selection criteria.

Input Command [I]

Specifies the primary input file.

INPUT *filename*

There can be only one Input file per task. The Input file should be created by Suprtool using the Output-Query or Output-Link option. If you want STExport to format date-fields and implied decimal places, you must use the Output-Link option of Suprtool when you create the file for input to STExport.

Every record in the input file is formatted into a corresponding record in the output file. It is best to have Suprtool Extract only the fields you actually need. Only those fields needed for import into the final application should be present in the Input file.

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 one as entered; 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 print the commands with ABSolute line numbers (the default), RELative line numbers (-5/-4), or UNNumbered. You can write the commands to your terminal. The Out option is not currently supported on STExport/UX. If you want to redo any of these commands, see Do, Redo, and Before.

Examples

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 @;rel     {print ALL, relative numbers}
$listredo rm        {print all rm commands}
$listredo rm xx     {print all "rm xx" commands}
$listredo @rm       {print all with "rm" anywhere}
```

Notes

The Listredo command cannot be abbreviated, but BJ is accepted as a short form.

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.

Output Command [O]

Specifies the Output file.

```
OUTPUT * | filename [ERASE] [LF] [NOLF]
```

By default, the Output file is named "Output". If you specify Columns None, the output file will have variable-length records. When Columns Fixed is specified, STExport creates the output file with fixed-length records.

Line Feeds

Stdlist

Line Feeds

STExport's Output command allows the user to specify whether the output file has line feeds. Normally STExport determines whether to write out line feeds from the self-describing file. If the self-describing file does not have line feeds, then the resulting file from the STExport task will not have line feeds. This caused problems for some programs that import the file or for some browsers that use the HTML option. You can now explicitly specify line feeds in the output file by using the LF option.

```
$in ora.customer  
$out filelex,lf  
$exit
```

To specify that line feeds are not written out to the file, you can use the NOLF option.

```
$in ora.customer  
$out filelex,nolf  
$exit
```

Stdlist

If the output file name is *, each output record is written to stdlist. This is useful for trying out different formatting combinations until you find the one that best fits the application that you want to import data into. For example,

```
$input      sdfilename
$output     *
$xeq
$floating   fixed 2      {change one option}
$input      sdfilename
$output     *
$xeq        {view the result}
$sign       none        {change a different option}
$input      sdfilename
$output     *
$xeq        {and so on}
```

Quote Command [Q]

Specify which quote character, if any, is to be used around byte-type fields.

QUOTE NONE | DOUBLE | SINGLE

(Default: Double)

Most software packages expect byte-type fields to be in one of two formats:

1. Fixed-column (see the Column command).
2. Surrounded by single- or double-quotes. In this case, you may also need to remove trailing spaces (see the Spaces command).

None

Double or Single

Fixed Columns (None)

Use Quote None to cause byte-type fields to be output as a group of characters. In many cases, you would combine this option with Columns Fixed.

Single or Double Quotes

By default, all byte type fields are surrounded by double quotes. Specify single quotes using the Single option. If a byte type field contains the quote character specified in the quote command, it is replaced with a space. For example, if the input was:

```
customer's
```

and Quote Single had been specified, then the output would be:

```
customer s
```

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)

The Redo command allows you to modify the commands before it executes them. If you do not need to change them, use the Do command. Commands are numbered sequentially from one as entered; 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

Examples

```
$ll fille      {'fille' is not spelled right}
fille not found
$redo         {redo most recent command}
ll fille      {last command is printed}
    d         {you enter changes to it}
ll file       {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"}
```

Reset Command [R]

Cancel the current task.

RESET

Reset closes the current Input file, then resets the Output file name to "Output". Formatting options are not reset, only the task-related commands are reset. If you try to reset an individual command, STExport prints a warning.

Set Command [S]

Enables or disables certain operating options within STExport. These options are not reset by Xeq or Reset commands.

```
SET  [MAPPED           ]  ON|OFF
      [STATISTICS       ]  ON|OFF
      [VARSUB           ]  ON|OFF
```

Mapped

Redo

Statistics

Variable Substitution

SET MAPPED ON (Default: OFF)

Mapped has no effect within STExport/UX.

Set Redo [*filename*]

(Default: none)
(Initially: temporary file)

Commands entered at the STExport 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. This temporary stack is not preserved across STExport invocations.

The new Set Redo command assigns a permanent file as the redo stack, allowing the stack to become available for future STExport invocations. For example, to assign the Myredo file as a persistent redo stack, enter

```
$Set Redo Myredo
```

If the file does not exist, STExport creates it. Otherwise, STExport uses the existing file. All subsequent commands are written to the persistent redo stack. The setting is valid for the duration of the STExport session. As soon as you exit STExport, the setting is discarded. Next time you run STExport, 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 STExport is using the default stack. Anything else is the name of the file used on the Set Redo command.

Concurrency
Qedit

Concurrency

When STExport uses the default temporary stack, it is only accessible to that particular instance of STExport. You can run as many STExport 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 only be used by one STExport 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, STExport 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 the command

```
set redo myredo
```

you will have a redo stack called Myredo for your STExport commands. If you exit STExport, then run Qedit and supply the same command Set Redo command, your Qedit commands will be written to the same file that was used for your STExport commands.

SET STATISTICS ON (Default: OFF)

Statistics causes STExport to print statistics at the end of each task.

SET VARSUB ON (Default: OFF)

Setting Variable Substitution causes STExport to resolve any CI variables in a command before processing.

Sign Command [SI]

Specify what should be done with the sign character for numeric-fields.

SIGN NONE | FLOATING | LEADING | TRAILING

(Default: Floating)

All numeric-type fields, except logical fields, have a sign. Integer and floating-point fields can have either a space " " (for positive values) or a negative sign "-". Packed- and display-type fields can have a space " " (neutral), a plus sign "+" (for positive values), or a negative sign "-".

Specify Sign None to cause STExport to completely ignore the sign. If you specify Sign None, no error or warning message appears if any numeric-types have a negative value.

Leading vs. Floating
Trailing

Leading vs. Floating

If you specify Columns Fixed, it is easy to see the difference between a leading versus a floating sign. A leading sign always appears in the same column whereas a floating sign always appears before the first digit of a number. For example,

Sign Leading	Sign Floating
- 22415	-22415
- 207	-207
- 16600	-16600
- 21910	-21910
- 8411	-8411
- 42	-42
- 16713	-16713
- 7970	-7970

Trailing Sign

Specify Sign Trailing to cause the sign character to appear after each formatted number. Remember that for many numeric-types the sign for positive numbers is a space. STExport always leaves room for the sign, even if it is a space.

Spaces Command [SP]

Specify whether trailing spaces are to appear in byte-type fields.

SPACES NONE | TRAILING

(Default: None)

If byte-type fields are surrounded with quotes (see the Quote command), the Spaces command determines whether trailing spaces in byte-type fields appear within the quotes. Use Spaces Trailing if you want to retain all of the spaces in a byte-type field.

Software packages that store variable-length character data treat trailing spaces as data. Use Spaces None to remove trailing spaces for data that is imported into these applications.

Use Command [U]

Specifies a file of commands to be executed as a group.

USE[Q] *filename*

Examples

Notes

Examples

A usefile makes your task easier by allowing common commands to be specified once in an external file. For example, the following usefile contains all the commands for creating the Invcust file:

```
$use invuse
input    invoices           {input file to format}
floating fixed 2           {formatting option}
output   invdata           {produces the file we want}
exit
```

STExport prints the lines in the usefile, including the comment lines. This allows you to include instructions and reminders in the usefile. In the example above, there were no commands for the user to enter.

Notes

Usefiles cannot be nested in STExport. The usefile may be any unnumbered text file or a Qedit workfile, but no more than 256 characters per record are processed.

By default, STExport displays the commands in a usefile as they are executed. STExport can execute commands *quietly* using the Useq command. For compatibility with Qedit, Useq can be abbreviated to UQ.

Verify Command [V]

Print the definition of the current task.

VERIFY

Verify prints the current Input and Output files and all export specifications; in other words, it is a Verify All command.

Xeq Command [X]

Perform the current task.

XEQ

Xeq checks that you have specified an Input file and an Output file. Then it performs the task and creates the Output file. Finally, it closes the files, ready for you to specify another task or Exit. If you also wish to leave STExport after completing the task, use Exit instead of Xeq.

Zero Command [Z]

Specify whether leading zeros are to appear in numeric fields.

ZERO NONE | LEADING

(Default: None)

Use Zero None to force all numeric fields to have leading zeros removed. If a numeric field has implied decimal places, STExport always formats the number with at least one digit to the left of the decimal place, even if it is zero.

Use Zero Leading to force all numeric fields to be zero-filled. In this case, Sign Leading and Sign Floating both cause the sign to appear in the same place (in front of the leading zeros).

Example of STExport Output

In this example we show you how to use Suprtool and STExport. We start with an SQL table, identify the fields that are dates and the number of implied decimal places in other fields. We then produce a self-describing file using the dataset as input and show the default output from STExport.

The Form command displays the fields in a dataset or a self-describing file. For files, this information is stored in a file with an extension of ".sd" and is not accessible with other tools. Use the Form command to obtain the record layout of STExport input files.

Sales File

Dates and Decimal Places

Salefile

STExport Output

Sales File

We will be formatting data from an Oracle table that has the following form:

```
>op oracle custdb dbpass
>sel * from sales
>form
Column Name:          Oracle Type:          Suprtool Type:

CUSTOMERNUM           Number   (8)           Double
DELIVERYDATE          Number   (8)           Double
PRODUCTNUM            Number   (8)           Double
PRICE                  Number   (10)          Packed
PURCHASEDATE          Number   (8)           Double
SALESQTY               Number   (10)          Packed
SALESTAX               Number   (10)          Packed
SALESTOTAL            Number   (10)          Packed
```

Dates and Decimal Places

We use Suprtool's Define and Item commands to identify which of the fields in the sales table are dates and which fields have implied decimal places:

```
>def deliv_date,DELIVERYDATE,8,display
>def purch_date,PURCHASEDATE,4,double
>def product_price,PRICE,4,double
>def sales_tax,SALESTAX,4,double
>def sales_total,SALESTOTAL,4,double
>item deliv_date    ,date    ,yyyymmdd
>item purch_date    ,date    ,yyyymmdd
>item product_price ,decimal ,2
>item sales_tax     ,decimal ,2
>item sales_total   ,decimal ,2
```

Salefile

We now produce a file called "salefile" using Suprtool's Output,Link option. The Link option produces a self-describing file, complete with the date and decimal-place information:

```
>open oracle custdb dbpass
>select * from sales
>def deliv_date,DELIVERYDATE,8,display
>def purch_date,PURCHASEDATE,4,double
>def product_price,PRICE,4,double
>def sales_tax,SALESTAX,4,double
>def sales_total,SALESTOTAL,4,double
>item deliv_date      ,date      ,yyyymmdd
>item purch_date     ,date      ,yyyymmdd
>item product_price  ,decimal  ,2
>item sales_tax      ,decimal  ,2
>item sales_total    ,decimal  ,2
>extract customernum
>ext deliv_date
>ext PRODUCTNUM
>ext product_price
>ext purch_date
>ext SALESQTY
>ext sales_tax
>ext sales_total
>output salefile,link
>xeq
IN=8, OUT=8. CPU-Sec=1. Wall-Sec=1.

>form salefile
  File: salefile      (SD Version B.00.00)  No linefeeds
  Entry:              Offset
  CUSTOMERNUM        I2      1
  DELIV_DATE         Z8      5 <<YYYYMMDD>>
  PRODUCTNUM         I2      13
  PRODUCT_PRICE      I2      17 << .2 >>
  PURCH_DATE         I2      21 <<YYYYMMDD>>
  SALESQTY           P12     25
  SALES_TAX          I2      31 << .2 >>
  SALES_TOTAL        I2      35 << .2 >>
  Entry Length: 38  Blocking: 1
```

Notice how the Form command correctly identifies which fields are dates and which fields have implied decimal places. STExport uses this information to format the file.

STExport Output

We then use STExport to read the self-describing Salefile to produce our sample output on stdlist.

To demonstrate how dates are handled, we insert a separator in each date field:

```
/opt/robelle/bin/stexport
$input salefile      {self-describing input file}
$date yyymmdd "-"    {dates with a dash separator}
$output *            {output to stdlist}
$xeq
10020,1993-10-05,50511501,98.31,1993-10-01,0.02,27.53,224.15
10003,1993-10-15,50511501,98.31,1993-10-15,0.01,13.76,112.07
10003,1993-10-15,50512501,145.62,1993-10-15,0.01,20.39,166.00
10003,1993-10-15,50513001,192.20,1993-10-15,0.01,26.91,219.10
10016,1993-10-21,50521001,24.59,1993-10-21,0.03,10.33,84.11
10016,1993-10-21,50532001,139.85,1993-10-21,0.01,19.58,159.42
10020,1993-10-28,50512501,146.60,1993-10-28,0.01,20.52,167.13
10010,1993-10-20,50533001,69.92,1993-10-20,0.01,9.79,79.70
```

There are no byte-type fields in the input file, so all fields are converted from their internal numeric representation to a string of digits. All date fields were converted from their internal yyymmdd format to the external yyymmdd format with a dash separator between day, month, and year. All fields with implied decimal places have been converted with a decimal point.

Limits Within STExport

Delimiter
Input File
Output File

Delimiter - Maximum Length - 3 Bytes

The delimiter must appear between every field in the output record. To help avoid exceeding the maximum output record length, the maximum delimiter length is three characters.

Input File - Maximum Record Size - 4096 Bytes

We recommend that you use Suprtool's Extract command to minimize the input record size.

Input File - Maximum Fields - 255

If you must have more than 255 fields, use Suprtool's Define and Extract commands to extract several fields as one contiguous series of bytes.

Output File - Maximum Record Size - 4096 Bytes

When formatting many fields, it is possible to produce large output records. Once again, using the Extract command to minimize the size of the input records will avoid large output records.

The total length of the Heading line in the output file is also restricted to 4096 bytes.

STExport

STExport reads a self-describing input file and applies formatting rules to create an output file, which is suitable for importing into other applications. STExport accepts only one command per line, but commands can be continued on the next line with an ampersand (&). The STExport prompt character is "\$", not ">."

`/usr/robelle/bin/stexport`

[Before](#)

[Columns](#)

[Date](#)

[Decimal](#)

[Delimiter](#)

[Do](#)

[Exit](#)

[Floating](#)

[Form](#)

[Heading](#)

[Help](#)

[HTML](#)

[Input](#)

[Listredo](#)

[Output](#)

[Quote](#)

[Redo](#)

[Reset](#)

[Set](#)

[Sign](#)

[Spaces](#)

[Use](#)

[Verify](#)

[Xeq](#)

[Zero](#)

Before see Suprtool section

Columns **FIXED** | **NONE**

Specify whether fields are formatted into variable- or fixed-length columns.

DAte **NONE** | **date-format** [**"separator"**]
| **INVALID** [**ASTERISKS** | **NULL** | **"string"**]

Specify a date format for all dates and how to process invalid dates.

DECimal PERIOD | COMMA

Specify which symbol will be used to indicate the decimal place in numeric fields.

Delimiter NONE | COMMA | TAB | "string"

Specify which delimiter character, if any, is to appear between each formatted field.

DO see Suprtool section

Exit [**ABORT** | **SUSPEND** | **XEQ**]

Perform specified task and return to HP-UX. The **Suspend** option is only available in Suprtool for MPE.

`$exit` {default = Xeq}

FLoating **DEFAULT** | **FIXED decimal-places** |
SCIENTIFIC decimal-places

Specify the format and the number of decimal places for floating-point fields.

Form [**filename**]

Display the fields in a self-describing file.

```
$form {default = show fields in the input file}
$form customer
```

HEAding NONE | FIELDNAMES | "string" |
ADD "string" | COLUMN "string" |

Specify which heading, if any, is to appear as the first record of the output file.

HElp [**command-name** | **keyword** [**,section**]]

Access the on-line user manual (/usr/robelle/help/stexport).

\$help {default is browse all}

\$help input {explain Input command}

HTML NONE | PREFORMATTED | TABLE
TITLE "string"
HEADING "string"

Use HTML to format Web pages for either Internet or intranet applications.
`$html table title "Product Listing"`

Input filename

Select a file to read and format.

```
$input invoices
```

LISTREDO see Suprtool section

Output * | **filename** [**LF**] [**NOLF**] [**ERASE**]

Specify output file name and whether it is temporary.

\$output custsale {default file name = Output}

Quote NONE | DOUBLE | SINGLE

Specify which quote character, if any, is to be used around byte-type fields.

REDO see Suprtool section

Reset

Cancel the current task.

```
$reset {reset everything}
```

Set option-name value

Set configurable options.

```
$set mapped on
```

```
$set statistics on
```

```
$set varsub on {MPE/iX only}
```


Sign NONE | FLOATING | LEADING | TRAILING

Specify what should be done with the sign character for numeric fields.

SPaces **NONE** | **TRAILING**

Specify whether byte-type fields have trailing spaces.

Use[Q] filename

Execute commands from a Text or Qedit file.

```
$use cap2.infile
```

Verify

Show current specifications.

```
$verify {show everything}
```

Xeq

Perform the current STExport task.

\$xeq

Zero NONE | LEADING

Specify whether numeric fields have leading zeros.

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: toth@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

