L4X v2.0 Reference

(v2.0.0.4 - 01 December 2008)

http://www.pdfing.com/l4x

Table of Contents

Foreword

An L4X script directs how L4X filters text into a copy of the worksheet containing the script. The L4X scripting language extends Lua 4.0 with reserved tables that define how text strings may be conditionally selected and converted before output to a formatted Excel worksheet. This document introduces the L4X language and also describes the most significant features of the Lua 4.0. language.

1 L4X Language Summary.

An L4X script specifies neither the name of the text-file nor the name of the workbook. These input and output files are automatically opened by the L4X filter when the script is run. The simplest possible L4X script directs the filter to read all (non-blank) lines from the input text-file and write them to successive cells of column **A** in the output worksheet:

```
-- An L4X script must begin with a comment (--) line
-- No other language statements are required!
```

Of course, L4X can perform more complex imports. The following script selects the first **10** text characters, from all lines of text in the input file, and writes them to successive cells of column **A**, beginning at row **1** of the output worksheet:

```
-- begin each script with a comment line
Column.A = {pos=1,len=10}
```

A detailed discussion of statements that import strings of text is provided in the next section of this document.

A script may specify that the text-strings are read only from selected lines within selected pages. The following script selects lines of text beginning at line **10** from every page of the input file, and writes text-strings read from the selected lines to successive cells of columns **A** and **B** beginning at row **3** of the output worksheet:

```
-- another script
Column.Source = {fromline=10, toline=-1, frompage=1, topage=-1}
Column.A = {pos=1  ,len=10}
Column.B = {pos=21, len=20}
Sheet.Out.row = 3 -- the "start" row
```

The last statement in the script above assigns a value to field: **`row`** of the Sheet.Outtable which specifies the **first row** of the worksheet where text read by **`Column.A`** and **`Column.B`** will be written.

The following script, provided for tradition's sake, writes "hello, world" to the output device:

```
-- all language manuals show how to do this!
print("hello, world")
```

1.1 Importing Strings of Text.

To read one string of text and write it to one worksheet cell, a script must include a statement that specifies both the cell location in the worksheet where text is to be written and also the position and length of the text-string to be read. For instance:

```
Cell.A1 = {pos=1, len=15, line=1, page=1, fmt="number"}
```

This statement adds a new field named: **A1** (which specifies worksheet cell **A1** as the write location) to the "reserved" table: Cell and assigns a new table (specifying the text-string to be read) to the new field: **`Cell.A1`**. Within the table constructor, the **`pos`** field specifies where the string starts within the line of text, the **`len`** field specifies how long it is, the **`line`** field specifies the line of text within the page and the **`page`** field specifies the page of text within the text-file. The **`fmt`** field specifies how the text-string is converted before it is written. A detailed analysis of a similar statement is provided in a following section of this document.

To read strings from successive lines of text and write them to successive cells of a worksheet, a script must include statements like:

```
Column.B = {pos=20, len=30}
```

This statement adds a new field named: **B** (which specifies worksheet column **B** as the write location) to the reserved table: Column and assigns a new table (specifying the text-strings to be read) to the new field: **`Column.B`**. Within the table constructor, the **`pos`** field specifies where the string starts within the line of text and the **`len`** field specifies how long it is. The table constructor includes neither field: **`line`** nor field:**`page`**, because the range of text lines to be selected is specified by the Column.Source table. A detailed analysis of a similar statement is provided in a following section of this document.

When a script adds a new field to the Column table its name does not specify an output row within the column, because this first row is specified by field: **`row`** of the**`Sheet.Out`** table. The filter adds a new row to the output worksheet beginning at the first row, for each line of text that is selected**.**

1.2 Importing Delimited Values

An L4X script can describe the import of one or more "delimited" values from within a single text-string. To do this, the table constructed and assigned to a new field of the Column table must assign a number to the **`value`** field. This number specifies the sequence-number of a single delimited value within a text-string, where **`1`** is the first delimited-value. When the **`value`** field is set but **`len`** is not, then the text-string containing the delimited values starts at the specified **`pos`** position and ends at the end of the text-line.

```
-- Select the first, second and third delimited values in a string
Column.C = {pos=1, value=1}
Column.D = {pos=1, value=2}
Column.E = {pos=1, value=3}
```

The default delimiter: **,** (comma) is specified in field: **`delimit`** of the Sheet.Pictures table

```
-- Change the value of the delimiter to the tab character
Sheet.Pictures.delimit  = "*tab"
Sheet.Pictures.quote    = "\"" -- the default quote character
```

Please note that, when changing the value of one field in the **`Sheet.Pictures`** table, a script need not construct and assign a new table. Once a table has been constructed and assigned a script can refer to one of its fields using the **.** index by name operator and assign a value directly to this field.

1.3 Converting Text Strings

If a script writes text to cells which are formatted as numbers or dates, these text-strings must be converted into numbers. For such cases the script should specify a value for the **`fmt`** field of Cell and Column tables, which specifies how text is to be converted into a number.

```
Column.A = {pos=10, fmt="number"}
```

When L4X converts text-strings to numbers or dates, it uses a corresponding "picture" to control the conversion. The reserved table Sheet.Pictures contains field names corresponding to each permitted value of the **`fmt`** field. Each of these "picture" fields contains a string that specifies the characters that stand for decimal-points, minus-signs, date and time separators and the ordering of date and time sub-strings. A script may change the "picture" in order to localize the conversion, as in the following example:

```
-- uses picture-string in Sheet.Pictures.date
Column.A = {pos=10, fmt="date"}
Column.B = {pos=20, fmt="date"}
Column.C = {pos=30, fmt="number"}
-- changes the picture-string for all dates
Sheet.Pictures.date = "yy/mm/dd"
```

Please note that, when changing the value of one field in the **`Sheet.Pictures`** table, a script need not construct and assign a new table. Once a table has been constructed and assigned a script can refer to one of its fields using the **.** index by name operator and assign a value directly to this field.

If only one column needs to be converted using a particular "picture", then it may be assigned to the **`picture`** field of the table that specifies the string to be read:

```
-- overrides Sheet.Pictures.date for column A only
Column.A = {pos=10, fmt="date", picture="yy/mm/dd"}
```

1.4 Selecting Lines of Text

When fields **A** ... **IV** are added to Column table by a script, this directs the L4X filter to write a worksheet cell for each (non-blank) line of text it reads. A script can select which lines of text are read by assigning values to fields of the Column.Source table. The following statement restricts column processing to the lines of text on or after line **10** in every page of the text-file.

```
Column.Source = {fromline=10, toline=-1, frompage=1, topage=-1}
```

The negative value for field **`toline`** specifies a line number relative to the last line in the page, where **-1** is the last line, **-2** is the next to last and so on. The negative value for field **`topage`** specifies a page number relative to the last page in the file, where **-1** is the last page, **-2**, is the next to last and so on.

Text-files sometimes "fold" a row of related text-strings into more than one line of text. In such cases, a script can specify a value for field: **`linesperrow`** of the **`Column.Source`** table:

```
Column.Source.linesperrow = 4
```

Here, the number assigned to field **`linesperrow`** specifies that four lines of text will be read before a worksheet row is written. If a script requires the import of strings from the second or subsequent lines of text, a script must assign an appropriate value to field **`lineofrow`** for each such import, like so:

```
Column.A = {pos=1, len=10, lineofrow=2}
Column.B = {pos=1, len=10, lineofrow=3}
Column.C = {pos=1, len=10} -- default lineofrow value = 1
```

1.5 Excluding Lines of Text.

The L4X filter always excludes blank lines of text. A script may also specify that, when a blank line is detected, all remaining lines in the current page are excluded or all remaining lines in the text-file are excluded.

```
-- on blank line - skip to next line (Default).
Column.Source.onblankline = "ignore"
```

```
-- on blank line - skip to next page.
Column.Source.onblankline = "endpage"
-- on blank line - skip to end-of-file.
Column.Source.onblankline = "endtext"
```

Please note that, when changing the value of one field in the **`Column.Source`** table, a script need not construct and assign a new table. Once a table has been constructed and assigned a script can refer to one of its fields using the **`.`** index by name operator and assign a value directly to this field.

To exclude a non-blank text line from within the range selected for reading, an L4X script should define the event function Event.OnOpenRow() and, within this function, assign a negative number to field: **`Column.Source.condition`** which excludes one or more lines of text:

```
function Event.OnOpenRow(Page, Source)
    -- exclude the current line
    if  (strsub(Page.text, 81, 85) == "-----") then
        Source.condition = -1
        end
    -- exclude the current line and next line
    if  (strsub(Page.text, 81, 85) == "Total") then
        Source.condition = -2
        end
    -- exclude current and following lines on the current page
    if  (strsub(Page.text, 81, 85) == "=====") then
        Source.condition = -9999
        end
   end
```

The Lua **`strsub()`** function is described in the Lua Functions section of this document. Please see the following section for a detailed discussion of event functions.

1.6 Event functions

When input text is not regularly formatted, a script may define functions for fields of the event table. An event function can inspect the current contents of the line and page of text which is being processed and control which columns can import text. A script may define a function for field Event. OnOpenRow, which is called,automatically, by the L4X filter before each group of text-lines (selected for column processing) is processed. A script may also define a function for field Event.OnOpenPage , which is called, automatically, by the L4X filter, before each page of text is processed.

The first function parameter (**`Page`**) passed to event functions, is a reference to the Sheet.Page table, the second function parameter (**`Source`**) is a reference to the Column.Source table. These parameters are passed so that expressions in event functions can refer to fields of these tables by the short names **`Page`** and **`Source`**, aiding clarity and convenience. An event function definition in L4X looks like:

```
function Event.OnOpenRow(Page, Source)
-- this function does nothing !!
end
```

Because the examples of event functions in this section inspect various fields of the Sheet.Page table, a brief description of these fields is given below:

```
Page.text          -- text from line in process.
Page.texts[]      -- array of text lines in page.
```

If a script needs to exclude particular lines of text, depending on the contents of the text currently being processed, a script may define a function for field Event.OnOpenRow that performs the necessary tests. For example:

```
-- define an event function
function Event.OnOpenRow(Page, Source)
if  (strsub(Page.text, 81, 85) == "=====") then
     -- exclude this (and the following) line.
     Source.condition = -2
     end
end
```

Here, if the current line of text contains the characters **"======"** at position **81** through **85**, the script will set the value of field: Column.Source**.condition** to **-2** (minus two). When the value of field **`Source.condition`** is a negative number then that number of lines, starting at the current line, are excluded from column processing. The value of field **`Source.condition`** is always set to zero before event functions are called, so a function does not need to set the zero condition explicitly.

If a script needs to adjust the start line for column processing, a script may define a function for field Event.OnOpenPage. For instance, when a text-file has "header pages" with fewer detail lines than succeeding pages, a script may define a function like:

```
-- lines on the "header page".
Column.Source = {fromline=20, toline-1}
Column.Heading.A = {pos=20, len=30, line=5}
Column.B         = {pos=1, len=10}
-- define an event function
function Event.OnOpenPage(Page, Source)
if  (strsub(Page.texts[5], 1, 8) ~= "Library:") then
     -- This is not a "header" page.
     Source.fromline = 10
     end
```

Here, the script tests a sub-string of the text in line **5** of the current page. When the sub-string "Library:" is not found, the range of lines selected for column processing is adjusted by assigning a different value to field **`Column.Source.fromline`**. Note the use of **`[]`**, the indexing operator and **`~=`,** the inequality operator. The fields **`Source.fromline`** and **`Source.toline`** are always reset to their initial values before event functions are called, so a function does not need to re-set them explicitly.

1.7 Conditional Import of Text.

An L4X script can **conditionally** control the way in which text is imported. This control is made possible by the **`condition`** field of the tables: Column.Format, Column.Heading and Column.Total. These tables will only perform text import when the value of their specified **`condition`** field matches any one of the conditions set in the Event.OnOpenRow() function. If the event function executes the following statement:

```
Source.condition = 1
```

It sets the **condition** to **1**. Please note that when a script assigns a **non-zero** number to this field, **none** of the locations specified by field names **A ... IV** of the Column table will be written to, nor will a new row be added to the worksheet.

If the event function executes the following statements:

```
Source.conditions[1] = 2
Source.conditions[9] = 3
```

The first of the statements above, assigns a number to array element **1** of field **`Source.conditions`** setting one of the conditions to **2**. The second statement assigns a number to array element **9** of field **`Source.conditions`** setting another of the conditions to 3. Although these statements set conditions, **all** of the locations specified by field names **A** ... **IV** of the Column table will be written to and a new row will be added to the worksheet.

If the text read for a worksheet column is already sorted, a script can set conditions when the value in a column changes, as follows:

```
Column.A = {pos=1, len=10, conditionset=3}
```

The statement above assigns a number to field Column.A**.conditionset** which specifies the conditions which are set when the value of the text-string selected for Column.A changes. Here, conditions **1**, **2**, and **3** are set when the value of the selected text-string is changed. Although this statements sets conditions, **all** of the locations specified by field names **A** ... **IV** of the Column table will be written to and a new row will be added to the worksheet.

1.8 Repeated Import of Text

The L4X filter can be directed to read a single text-string but write it to many worksheet cells. If a script requires L4X to read a text-string once per page but write it **repeatedly** to succeeding cells of a worksheet, a script may assign a new table to the appropriate field of the Column.Heading table. A script may include a statement like:

```
Column.A         = {pos=1, len=30}
Column.Heading.B = {pos=1, len=30, line=5}
```

Here, the text-string written to column **B** is read from line **5** of every page, provided that line **5** is **not** within the range of lines lines specified by the Column.Source table. This text-string is then written to successive cells of column **B**, whenever text is written to Column **A** and a new row is added to the worksheet.

If a script requires L4X to read text-string from a line within the range of lines specified by the Column.Source table, but write it **repeatedly** to **succeeding** cells of a worksheet, a script may include a statement like:

```
Column.A         = {pos=1, len=30}
Column.Heading.B = {pos=10, len=30, fmt="text", condition=1}
```

Here, the text-string written to column **B** is read from position **10** of each line within the lines specified by the **`Column.Source`** table, provided that one of the conditions is set to **1**. This text-string is repeatedly written to successive cells of column **B**, whenever text is written to Column **A** and a new row is added to the worksheet.

If a script requires L4X to read a text-string from a line within the range of lines specified by the Column.Source table, but write it **repeatedly** to **preceding** cells of a worksheet, a script may assign a new table to the appropriate field of the Column.Total table. A script may include a statement like:

```
Column.A       = {pos=1, len=30}
Column.Total.C = {pos=99, len=10, fmt="number", condition=2}
```

Here, the text-string written to column **C** is read from position **99** of each line within the lines specified by the **`Column.Source`** table, provided that one of the conditions is set to **2**. This text-string is then written to preceding cells of column **C** which have not previously been written.

1.9 Formatting Cells and Columns.

An L4X script does not directly control the appearance and formatting of the output worksheet. However the formatting and formulae of the workbook containing the script are preserved in the copy of the "scripted" workbook output by the L4X filter. Any references in cell formulae, ranges and charts are adjusted to allow for the rows of text imported by the filter, as are the relative positions of notes, word-art and pictures. In particular the formats and formulae of all the cells in the first row of the worksheet are automatically copied and adjusted before each new row is imported.

A script may also copy and adjust formats and formulae from one area of the worksheet to another by adding fields to the reserved table: Column.Format. This makes it possible to insert a row containing formulae that calculate (for instance) sub-totals. In the following example script:

```
-- define columns
Column.A = {pos=1,  len=10, conditionset=1}
Column.B = {pos=20, len=15, fmt="number"}
-- Insert formatting and formula
Column.Format.B = {condition=1, fromcell="Z1"}
```

When the condition is set to **1**, L4X will write the contents of cell **Z1** to column **B** of the worksheet. If cell **Z1** contains a formula like **=SUM(B1:B1)** the formula will be suitably adjusted each time it is written. So, when the formula is written to row **8** it becomes: **=SUM(B1:B8)**, when written to row **9** it becomes: **=SUM(B1:B9)**, and so on.

1.10 Changing Worksheet properties

Fields of the reserved table Sheet.Out allow a script to specify certain properties of the worksheet output by L4X. The L4X filter will, by default, protect the output worksheet.

A script can allow the output worksheet's contents to be edited, using statement:

```
Sheet.Out.protect = 0 -- allow editing
```

Please note that if the scripted worksheet is protected, then the output worksheet cannot be unprotected by L4X.

L4X will also, by default, remove any VBA macros in the output workbook. VBA macros can be saved, by including the statement:

```
Sheet.Out.vbsave = 1 -- save VBA macros
```

1.11 Analysis of an L4X statement

The most common statements in an L4X script describe the text-string(s) to be read from an input text-file and the worksheet cell or cells where they are written. A typical statement looks like:

```
Column.B = {pos=1, len=15, fmt="number"}
```

All assignments to reserved tables **`Cell`** and **`Column`** follow the same pattern. The field name added to the **reserved table** (on the left-hand side of the assignment operator =) specifies the worksheet location where text is written, and the value assigned to this field (on the right-hand side of operator =) is a reference to a table which specifies the text-string(s) to be read from the input text-file. Each element of this statement is described, in detail, below:

Column
names one of the L4X reserved tables. A script may add new fields to this table, where the field name must specify a column location within a worksheet.

.
this period is the index by name operator which is used to add a field named **B** to the **Column** table.

B
is the name of the field added to the Column table. The name specifies that text will be written to column **B** of the worksheet. This name does not specify an output row, because its first row is specified by field: **`row`** of the **`Sheet.Out`** table.

=
is the assignment operator. It assigns a value to the field **Column.B**. The value assigned is specified on the right-hand side of the **=** operator and this value must be a reference to a table, the fields of this table specify which text-strings are read from the input file.

{
is the start of the table constructor operator. A script may specify field names and initialize their values within the **table constructor**.

pos=1
adds a field named **pos** to the table being constructed and initializes its value to **1**. This field specifies the left-most position of a text-string within a line of text.

,
this comma separates each element in the list of **fieldname=value** pairs within the **table constructor**.

len=15
adds a field named **len** to the table being constructed and initializes its value to **15**. This field specifies the length of the text-string which begins at position **1**.

,
this comma is another list-separator.

fmt="number"
adds a field named **fmt** to the table being constructed and initializes its value to **"number"**. This field specifies how the text-string is to be converted before it is imported.

}
is the end of the table constructor operator.

1.12 Lua Tables and Types

Lua 4.0 is a dynamically typed language. This means that variables do not have fixed types, but the value assigned to a variable will have one of the Lua types: **nil**, **number**, **string**, **function**, **userdata** and **table**.

A value of type **nil** is returned by any undefined variable name, a **false** result from the relational and logical operations or may be returned by a call to a function as described in following sections of this document. A script may test for a **nil** value as follows:

```
 if  (not x) then
```

```
        -- x is nil
        end
```

A value of type **table** is a reference to a table object, constructed by the Lua operator **{ }**. A Lua table can contain many fields, each field having a unique name and an associated value. Because a Lua table is an "associative array", the values contained in a table can be indexed by number or name. Lua tables are constructed and used as follows:

```
   x = {}    -- use {} to construct a new empty table
   x.pos = 1 -- adds field-name pos with value 1 to table x
   x.len = 9 -- adds field-name len with value 9 to table x
```

For the sake of brevity, the table may be constructed and initialised in one expression, where each "field" **name=value** pair is delimited by a comma (**,**). The equivalent code is then:

```
   x = {pos=1, len=9} -- construct and initialise table
```

The value of an individual field of a table may also be referred to by its index, as follows:

```
   x["pos"] = 1
   x["len"] = 9
   print(x.len) -- >> 9
```

An **array** is a table with numeric field names. The following example demonstrates assignment to elements of an array, using the index operation:

```
   a = {}
   a[1] = "element one"
   a[2] = "element two"
   j = 1
   print(a[j]) -- >> "element one"
```

1.13 Lua Constants

String constants are delimited by matching single or double quotes, and can contain the C-like escape sequences:

- **\a** (bell)
- **\b** (backspace)
- **\f** (form feed)
- **\n** (newline),
- **\r** (carriage return)
- **\t** (horizontal tab)
- **\v** (vertical tab)
- \\ (backslash)
- **\"** (double quote)
- **\'** (single quote)

A character in a string may also be specified by its numerical value, through the escape sequence `\ddd', where ddd is a sequence of up to three decimal digits. Strings in Lua may contain any 8-bit value, including embedded zeros, which can be specified as `\000'.

Literal strings can also be delimited by matching **[[...]]**. Literals in this bracketed form may run for several lines, may contain nested **[[...]]** pairs, and do not interpret escape sequences. This form is especially convenient for writing strings that contain program pieces or other quoted strings. As an example, in a system using ASCII, the following three literals are equivalent:

```
1)   "alo\n123\""
2)   '\97lo\10\04923"'
3)   [[alo
     123"]]
```

Numerical constants may be written with an optional decimal part and an optional decimal exponent. Examples of valid numerical constants are:

- **3**
- **3.0**
- **3.1416**
- **314.16e-2**
- **0.31416E1**

Comments start with -- (two hyphens) and continue until the end of the line. Also, if the first line of a file starts with a # (hash), then the first line is also a comment.

1.14 Lua Operators

L4X scripts can use any of the operators defined in Lua 4.0.

- Assignment operator is: **=**

- Arithmetic operators are: **+ - / * ^**

- String concatenation operator is: ..

- Relational operators are:**== ~= < > <= >=**
  Relational operators return **nil** for false and any other value for true. Please note that ~= is the inequality operator.

- Logical operators are: **and or not**
  Logical operators consider **nil** as false and any other value as true. The conjunction operator and returns **nil** if its first argument is **nil**; otherwise, it returns its second argument. The disjunction operator or returns its first argument if it is different from **nil**; otherwise, it returns its second argument. Both and and or use short-cut evaluation, that is, the second operand is evaluated only if necessary.

- Constructor operator for tables and arrays is:**{}**

- Index by name operator for tables is: **.fieldname**
  This operator is equivalent to the index operator ["fieldname"] but is provided for convenience.

- Index operator for tables and arrays is: **[i]**
  Index value **i** may be a number or a string. An array is a Lua table without field names, so the expression **a[1]** refers to the first element in array **a** , **a[2]** refers to the second element, and so on.

1.15 Lua Control Structures

L4X scripts can use any of the control structures defined in Lua 4.0. For convenience**, if** is defined below as:

```
if logical-expression then
   statement(s)
end

if logical-expression then
   statement(s)
else
   statement(s)
end
```

The evaluation of a "logical expression" may return any value. All non-nil values are considered **true**, only the nil value is considered **false**.

Lua also provides **iterative** control structures **while** and **for**. L4X scripts should not require iteration and iterative control structures should be used with extreme caution.

1.16 Lua Functions

L4X scripts can use any of the functions defined in Lua 4.0 or in its standard libraries. The following functions from the Lua string manipulation library are documented here, for convenience, as they will often be used in event functions.

- **`strfind(s, pattern [, init [, plain]])`**
  Looks for the first match of pattern in s. If it finds a match, then strfind returns the indices of **s** where this occurrence starts and ends, otherwise, it returns **nil**. A third, optional numerical argument **init** specifies where to start the search, its default value is 1 and may be negative. A value of 1 as a fourth optional argument **plain** turns off the pattern matching facilities, so the function does a plain "find substring" operation with no characters in pattern being considered **magic**. Note that if **plain** is given, then **init** must be given too.

- **`strsub(s, i [, j])`**
  Returns another string, which is a substring of **s** starting at **i** and running until **j**. Both **i** and **j** may be negative. If **j** is absent, then it is assumed to be equal to -1 which is the same as the string length. In particular, the call strsub(s, 1, j) returns a prefix of **s** with length **j** and the call strsub(s, -i) returns a suffix of **s** with length **i**.

- **`print(s1, s2, ...)`**
  Prints the value of any number of parameters. The text output from this function will appear on the output page of the L4X form.

A script may, of course, define new functions:

```
function printhello() -- start function definition of "printhello"
    print("hello")
end                     -- end function definition
-- call function
printhello() -- >> "hello"
```

2 L4X Tables and Event Functions

The L4X filter automatically constructs reserved Lua tables that direct how text is imported. The names of these tables are chosen from the vocabulary of Excel so that L4X scripts can clearly express their purpose. The fields of reserved tables that select text-strings have names like: **`line`**, **`position`** and **`page`** for the same reason.

- Cell
  A script may add new fields to this table, where the field name must specify a **cell** location within a worksheet and the field value must be a table that describes how one text-string is read and converted.

- Column
  A script may add new fields to this table, where the field name must specify a **column** location within a worksheet and the field value must be a table that describes how a series of text-strings are read and converted. Other fields of the Column table are described below.

  - Column.Format
    A script may add new fields to this table, where the field name must specify a **column** location within a worksheet and the field value must be a table that describes when, where and how a cell (or series of cells) in a worksheet row are to be formatted.

  - Column.Heading
    A script may add new fields to this table, where the field name must specify a **column** location within a worksheet and the field value must be a table that describes how a series of text-strings are read and converted before being **repeatedly** written.

  - Column.Source
    This table contains fields that specify which pages and lines of text will be read by fields: **A** ... **IV** of the **`Column`** table. A script may replace the existing table with a new table or change the value of a one or more of its fields. For convenience, this table is passed as the **second** parameter of event functions added to the Event table.

  - Column.Total
    A script may add new fields to this table, where the field name must specify a **column** location within a worksheet and the field value must be a table that describes how a series of text-strings are read and converted before being **repeatedly** written.

- Event
  This table contains fields referring to the OnOpenPage, OnOpenRow and OnWriteX functions.

- Sheet
  This table contains fields that refer to the tables: Out, Page and Pictures.

  - Sheet.Out
    This table contains fields that specify miscellaneous parameter values relating to worksheet output. A script may replace the existing table with a new table or change the value of a one or more of its fields. The field **`Sheet.Out.row`** specifies the first row where fields **A** ... **IV** of the Column table will start writing and consequently controls the formatting of the output worksheet.

  - Sheet.Page
    This table contains fields that describe the current state of the filter. For convenience, this table is passed as the **first** parameter of event functions added to the Event table.

  - Sheet.Pictures

This table contains fields that contain the "picture" strings which control how a text-string may be converted before being written. A script may replace the existing table with a new table or change the value of a one or more its fields.

2.1 Cell.A1 ... IV65535

A script may write to a worksheet cell by constructing a table and assigning it to a new field of table **Cell**. This new field may be named **A1**, **A2** and so on, up to **IV65535**, where every cell in a worksheet has a corresponding field name. The table constructed and assigned to this new field specifies values for the following fields which control how a text-string is read and converted.

Fields of table: Cell.A1 ... IV65535		
Name	*Type*	*Description*
pos	positive-integer (mandatory)	Specifies the position of a text-string in a text line. If the text-string does not contain included blanks, a script can specify any position within the string and L4X will select the whole string bounded by blanks. If the text-string does contain included blanks then a script must specify the left-most position of the string and supply a value for the len field.
len	positive-integer (optional)	Specifies the length of a text-string in a text line. Used together with the pos field when the text-string contains included blanks. This field need not be specified for numbers, as long as the value of the pos field specifies the left-most number or the decimal point within the string.
line	integer (mandatory)	Specifies the line of the text-page from which the text-string is selected. A script may specify a negative number where -1 is the last line of a page, -2 is the next to last line, and so on.
toline	integer (optional)	If a script requires a worksheet cell to contain data from more than one line of text, a script must specify the last line of text from which the text-string is selected. A script may specify a negative number where -1 is the last line of a page, -2 is the next to last line, and so on.
page	integer (optional)	Specifies the page of the text-file from which the text-string is selected. A script may specify a negative number where -1 is the last page of a file, -2 is the next to last page, and so on. The default value is 1, the first page of the file.
fmt	string (optional)	Specify "number", "date", "time", "datetime", or "boolean" when a text-string must be converted to a numeric value. A script may also specify the default value "text" when no conversion is required, or "general", which provides automatic conversion to a numeric value when appropriate.
picture	string (optional)	If a script specifies a value for field fmt and the conversion picture in table Sheet.Pictures is not correct for this particular field then a script can specify an alternative picture string here.

The following example will read text from position **20**, line **4** and page **1** of the input text-file and write it to the output worksheet at location **A2**. The selected text will be converted to a number before being written.

```
Cell.A2 = {pos=20, line=4, page=1, fmt="number"}
```

2.2 Column

This table contains fields that define how text is read and the worksheet columns where text is written. Each field of this table is described in detail in following sections of this document.

Fields of table: Column		
Name	*Type*	*Description*
A ... IV	table	A script may add new fields to this Column table. The new field name must specify a worksheet column and its value must be a table that defines which text-strings are read and written to succeeding cells of the named column.
Format.A ... IV	table	Contains an empty table. A script may add new fields to this table, the new field name must specify a worksheet column and its value must be a table that defines which cells are copied to the specified worksheet column. All cell-references in the copied cell(s) that refer to the start-row wll be automatically adjusted to refer to an appropriate range of cells.
Heading.A ... IV	table	Contains an empty table. A script may add new fields to this table, the new field name must specify a worksheet column and its value must be a table that defines how text is read and repeatedly written to succeeding cells of the named column.
Source	table	Contains a reference to a table that defines the selection of text-lines.
Total.A ... IV	table	Contains an empty table. A script may add new fields to this table, the new field name must specify a worksheet column and its value must be a table that defines how text is read and repeatedly imported to preceding cells in the named column.

2.2.1 Column.A ... IV

A script may write to cells within a worksheet column by constructing a table and assigning it to a new field of the **Column** table. This new field may be named **A**, **B** and so on, up to **IV**, where every column in a worksheet has a corresponding field name. The name does not specify an output row within the worksheet column, because a new row is added to the output worksheet, starting at its first row, for each line of text that is selected. The table constructed and assigned to this new field specifies values for the following fields which control how text-strings are read and converted. Note that neither field **`line`** nor field: **`page`** are present in this table, because the range of lines from which text-strings are read is specified by values assigned to fields of the Column.Source table.

Fields of table: Column.A ... IV		
Name	*Type*	*Description*
pos	positive-integer (mandatory)	Specifies the position of a text-string in a text line. If the text-string does not contain included blanks, a script can specify any position within the string and L4X will select the whole string bounded by blanks. If the text-string does contain included blanks then a script must specify the left-most position of the string and supply a value for the len field.
len	positive-integer (optional)	Specifies the length of a text-string in a text line. Used together with the pos field when the text-string contains included blanks. This field need not be specified for numbers, as long as the value of the pos field specifies the left-most number or the decimal point within the string.
lineofrow	positive-integer (optional)	If the contents of one worksheet row is selected from multiple text-lines, a script may specify the sequence number of the line within the group of lines. The text for this column will be selected only when the specified line is reached. The default value is 1, the first text-line of the row.
fmt	string (optional)	Specify "number", "date", "time", "datetime", or "boolean" when a text-string must be converted to a numeric value. A script may also specify the default value "text" when no conversion is required, or "general", which provides automatic conversion to a numeric value when appropriate.
picture	string (optional)	If a script specifies a value for field fmt and the conversion picture in table Sheet.Pictures is not correct for this particular field then a script can specify an alternative picture string here.
trim	boolean (optional)	Specifies how string with leading blanks are written. 1 or "yes" will cause leading-blanks to be removed before import. Default value is "no", leading blanks are not removed before import.
blanktozero	boolean (optional)	Specifies how blank strings are converted to numbers. A value of 0 or "no" will cause blank strings to be ignored. Default value is "yes", blank strings are converted to zero before import.
conditionset	positive integer	Specifies the condition(s) to set when the value of the text-string changes. When the text-string changes, all positive conditions which are lower than the specified

	(optional)	number are also set. Default value is 0, no conditions are set.
value	number (optional)	Specifies the sequence-number of a delimited-value within the selected text-string. The default value is 0 (zero), not a delimited-value.
delimit	string (optional)	Specifies the character that is used to mark the end of this particular delimited-value. A script may use "*tab" to indicate that the delimiter is the tab character. The default value is the value specified in the Sheet. Pictures.delimit field.
quote	string	Specifies the character that is used to "quote" this particular delimited-value. A delimited-value may be quoted when it could itself contain a delimiter character. If a quote character appears within the string it must be "escaped" by a preceding quote character. The specified string may be zero characters long, where a particular delimited-value is not quoted. The default value is the value specified in the Sheet. Pictures.quote field.

The following example will write a column of cells at worksheet location **A**, reading **30** characters of text (containing included blanks) from position **40** of each line of text that is selected by the Column. Source table.

```
Column.A = {pos=40, len=30}
```

Text will only be written to the worksheet when the value of field **`Column.Source.condition`** is equal to **`zero`**.

2.2.2 Column.Format.A ... IV

A script may specify the contents of one or more worksheet columns, by constructing a table and assigning it to a new field of the **Column.Format** table. This new field may be named **A**, **B** and so on, up to **IV**, where every column in a worksheet has a corresponding field name. The table constructed and assigned to this new field specifies values for the following fields which control how and when a cell (or range of cells) is copied to the worksheet.

Fields of table: Column.Format.A ... IV		
Name	*Type*	*Description*
condition	positive-integer (mandatory)	Specifies the condition that causes cells to be copied.
fromcell	cell-reference (mandatory)	Specifies the position of a cell within this worksheet. The contents and formatting of all the cells in the range specified by the fromcell and tocell fields are copied to the named column in the specified row of the work sheet. Formulae in these cells that refer to the start-row are adjusted
tocell	cell-reference (optional)	Specifies the position of a cell within this worksheet. The contents and formatting of all the cells in the range specified by the fromcell and tocell fields are copied to the named column of this work sheet. The default value is the fromcell value.
at	string (optional)	Specifies where the specified cells are copied to. The string may be one of "start", "end" or "row". If "start" is specified then the specified condition signals the start of a set of rows and the specified cells are copied before the first row of the next set. If "end" is specified then condition signals the end of a set of rows and cells are copied onto or after the current row. If "row" is specified then condition signals one selected row and cells are copied onto the current row. The default value is "end"
functo	string (optional)	Specifies a column location to which (the first) adjusted formula in the selected cells will be copied. Can be specified only when at is "end". The default value is blank, formula is not copied.
insert	boolean (optional)	Specify 0 if a script requires the specified cells to be copied to the current row. Specify 1 if a script requires the cells to be copied to the next row. This value is automatically set to 0 for any "row" groups. The default value is 1 for "start" and "end" groups.
remove	boolean (optional)	Specify 0 if a script requires the specified cells to be retained in the output worksheet. Specify 1 if a script requires the cells to be removed from the output worksheet. The default value is 1, specified cells are removed.

The contents, **formats** and **formulae** of the specified cells are copied to the named column when a matching condition is set. If any of the copied formulae contain references to cells in the origin-row, then these cell references will be adjusted for the row to which the specified cell is copied.

The following example will copy the contents and format of the selected cells to the named column of the worksheet, **after** the current row.

```
Column.Format.B = {condition=1, fromcell="X1", tocell="Y1"}
```

The specified range of cells (**X1** to **Y1)** will copied to column **B** only when condition is set to **1**.

2.2.3 Column.Heading.A ... IV

A script may write to cells within a worksheet column by constructing a table and assigning it to a new field of the **`Column.Heading`** table. This new field may be named **A**, **B** and so on, up to **IV**, where every column in a worksheet has a corresponding field name. The table constructed and assigned to this new field specifies values for the following fields which control how a text-string is read and converted.

Fields of table: Column.Heading.A ... IV		
Name	*Type*	*Description*
pos	positive-integer (mandatory)	Specifies the position of a text-string in a text line. If the text-string does not contain included blanks, a script can specify any position within the string and L4X will select the whole string bounded by blanks. If the text-string does contain included blanks then a script must specify the left-most position of the string and supply a value for the len field.
len	positive-integer (optional)	Specifies the length of a text-string in a text line. Used together with the pos field when the text-string contains included blanks. This field need not be specified for numbers, as long as the value of the pos field specifies the left-most number or the decimal point within the string.
line	integer (optional)	Specifies the line of the text-page from which the text-string is selected. A script may specify a negative number where -1 is the last line of a page, -2 is the next to last line, and so on.
page	integer (optional)	Specifies the page of the text-file from which the text-string is selected. a script may specify a negative number where -1 is the last page of a file, -2 is the next to last page, and so on. The default value 0, selects the text on every page.
condition	positive-integer (optional)	Specifies the condition for reading a text-string. The default value 0 specifies un-conditional reading.
lineofrow	positive-integer (optional)	If the contents of one worksheet row is selected from multiple text-lines, a script may specify the sequence number of the line within the group of lines. The text for this column will be selected only when the specified line is reached. The default value is 1, the first text-line of the row.
fmt	string (optional)	Specify "number", "date", "time", "datetime", or "boolean" when a text-string must be converted to a numeric value. A script may also specify the default value "text" when no conversion is required, or "general", which provides automatic conversion to a numeric value when appropriate.
picture	string (optional)	If a script specifies a value for field fmt and the conversion picture in table Sheet.Pictures is not correct for this particular field then a script can specify an alternative picture string here.
trim	boolean (optional)	Specifies how string with leading blanks are written. 1 or "yes" will cause leading-blanks to be removed before import. Default value is "no", leading blanks are not removed

		before import.
blanktozero	boolean (optional)	Specifies how blank strings are converted to numbers. A value of 0 or "no" will cause blank strings to be ignored. Default value is "yes", blank strings are converted to zero before import.
value	number (optional)	Specifies the sequence-number of a delimited value within the selected text-string. The default value is 0 (zero), not a delimited value.
delimit	string (optional)	Specifies the character that is used to mark the end of this particular delimited-value. A script may use "*tab" to indicate that the delimiter is the tab character. The default value is the value specified in the Sheet. Pictures.delimit field.
quote	string	Specifies the character that is used to "quote" this particular delimited-value. A delimited-value may be quoted when it could itself contain a delimiter character. If a quote character appears within the string it must be "escaped" by a preceding quote character. The specified string may be zero characters long, where a particular delimited-value is not quoted. The default value is the value specified in the Sheet. Pictures.quote field.

Text-strings specified by fields of this **`Column.Heading`** table will be read, either when the value of the **`line`** field is outside the range defined by the Column.Source table or when a matching condition is set. The text-string will be written to the named column whenever a **Column** field adds a new row to the worksheet. In this way cells in a worksheet row can contain data from both "detail" and "heading" lines of a text file.

The following example will write text to the next cell in column **C** of the worksheet, when a new row is added to the worksheet.

```
Column.Heading.C = {pos=20, line=5}
```

The contents of the cells in column **C** will be read from position **20** of line **5** in every text-page, provided that the value of field **`Column.Source.fromline`** is greater than **5**.

2.2.4 Column.Source

The fields in this table control how the input text file is processed. Because tables assigned to fields **A** ... **IV** of the **Column** table specify values for neither field **`line`** nor field **`page`**, the text lines and pages to be selected are set for all new fields in table **`Column`** by specifying values for the equivalent fields in this **`Column.Source`** table.

Fields of table: Column.Source		
Name	*Type*	*Description*
fromline	positive integer (optional)	Specifies the starting text line of lines selected for column processing. Default value is 1, the first line on the page.
toline	integer (optional)	Specifies the ending text line of lines selected for column processing. A script may specify a negative number where -1 is the last line of a page, -2 is the next to last line, and so on. Default value is -1, the last line on the page.
frompage	positive integer (optional)	Specifies the starting text page of pages selected for column processing. Default value is 1, the first page in the file.
topage	integer (optional)	Specifies the ending text page of pages selected for column processing. A script may specify a negative number where -1 is the last page of text, -2 is the next to last page, and so on. Default value is -1, the last page in the file.
linesperrow	positive-integer (optional)	If the contents of one worksheet row is selected from multiple text-lines, a script must specify the number of lines that will be used to compose each row. The default value is 1.
onblankline	string (optional)	Specifies the action to be taken when a blank line of text is read. The string may be one of "ignore", "endpage" or "endtext". The default value is "ignore".
condition	integer (optional)	Used in event functions to set a condition. This activates any fields in the Column.Format, Column.Heading and Column.Total tables with the same number specified as the value of their condition field. Setting this value to a non-zero number will de-activate all Column.A .. IV tables. Setting this value to a negative number will cause the specified number of lines to be excluded from column processing. The value is set to zero, before event functions are called.
conditions	array (optional)	Used in event functions to set one or more conditions. This activates any fields in the Column.Format, Column. Heading and Column.Total tables with the same number specified as the value of their condition field. Any one of the nine elements of this array can be set, using the index operator [n], where n is an integer in the range 1 to 9. All element values in this array are set to zero, before event functions are called.

The following example specifies the lines of text that qualify for column processing. Additionally the value of field **linesperrow** specifies that one row of cells may have contents selected from two successive lines of text.

```
Column.Source = {fromline=10, frompage=1, linesperrow=2}
```

A script may set values for individual fields of this table in event functions The following code specifies values for elements of field conditions.

```
-- in event functions, Source refers to Column.Source
Source.conditions[1] = 1
Source.conditions[9] = 2
```

2.2.5 Column.Total.A ... IV

A script may write to cells within a worksheet column by constructing a table and assigning it to a new field of the **`Column.Total`** table. This new field may be named **A**, **B** and so on, up to **IV**, where every column in a worksheet has a corresponding field name. The table constructed and assigned to this new field specifies values for the following fields which control how a text-strings are read and converted.

Fields of table: Column.Total.A ... IV		
Name	*Type*	*Description*
pos	positive-integer (mandatory)	Specifies the position of a text-string in a text line. If the text-string does not contain included blanks, a script can specify any position within the string and L4X will select the whole string bounded by blanks. If the text-string does contain included blanks then a script must specify the left-most position of the string and supply a value for the len field.
len	positive-integer (optional)	Specifies the length of a text-string in a text line. Used together with the pos field when the text-string contains included blanks. This field need not be specified for numbers, as long as the value of the pos field specifies the left-most number or the decimal point within the string.
condition	positive-integer (mandatory)	Specifies the condition for reading a text-string.
lineofrow	positive-integer (optional)	If the contents of one worksheet row is selected from multiple text-lines, A script may specify the sequence number of the line within the group of lines. The text for this column will be selected only when the specified line is reached. The default value is 1, the first text-line of the row.
fmt	string (optional)	Specify "number", "date", "time", "datetime", or "boolean" when a text-string must be converted to a numeric value. A script may also specify the default value "text" when no conversion is required, or "general", which provides automatic conversion to a numeric value when appropriate.
picture	string (optional)	If a script specifies a value for field fmt and the conversion picture in table Sheet.Pictures is not correct for this particular field then a script can specify an alternative picture string here.
trim	boolean (optional)	Specifies how string with leading blanks are written. 1 or "yes" will cause leading-blanks to be removed before import. Default value is "no", leading blanks are not removed

		before import.
blanktozero	boolean (optional)	Specifies how blank strings are converted to numbers. A value of 0 or "no" will cause blank strings to be ignored. Default value is "yes", blank strings are converted to zero before import.
value	number (optional)	Specifies the sequence-number of a delimited value within the selected text-string. The default value is 0 (zero), not a delimited value.
delimit	string (optional)	Specifies the character that is used to mark the end of this particular delimited-value. A script may use "*tab" to indicate that the delimiter is the tab character. The default value is the value specified in the Sheet. Pictures.delimit field.
quote	string	Specifies the character that is used to "quote" this particular delimited-value. A delimited-value may be quoted when it could itself contain a delimiter character. If a quote character appears within the string it must be "escaped" by a preceding quote character. The specified string may be zero characters long, where a particular delimited-value is not quoted. The default value is the value specified in the Sheet. Pictures.quote field.

Text-strings specified by fields of this **`Column.Total`** table will be read whenever a matching condition is set. The text-string will be written to the (relevant) preceding cells of the named column. In this way cells in a worksheet row can contain data from both "detail" and "total" lines of a text file.

The following example, will write a text-string to all the preceding cells in column **D** of the worksheet. If a preceding cell has previously been written, then this cell is not changed.

```
Column.Total.D = {pos=20, condition=1}
```

The contents of the cells in column **D** will be read from position **20** of every line, provided that a condition is set to **1**.

2.3 Event

The fields of this table contain the definitions of event functions. A script should define these functions when the input text file is not regularly formatted, as they allow a script to examine the imported text at run time and exclude lines of text or condition which columns will import text-strings.

Fields of table: Event		
Name	***Type***	***Description***
OnOpenPage	**function** **(optional)**	A script may define a Lua function and assign it to this field of this **`event`** table. If a script defines this function, it will be called automatically before each page of the text-file is processed.
OnOpenRow	**function** **(optional)**	A script may define a Lua function and assign it to this field of this **`event`** table. If a script defines this function, it will be called automatically before each line of the text-file is processed.
OnWriteX	**function** **(optional)**	A script may define a Lua function and assign it to these fields of this **`event`** table, where **X** is a valid column location. If a script defines a function for any particular column, it will be called automatically before text is written to cells in the specified column.

Any event functions that a script defines are called automatically, when the script is run. A script should **not** call these functions explicitly. The functions that may be assigned to the fields of the **Event** table are described, in detail, below.

2.3.1 Event.OnOpenPage

This field of the Event table may contain a function definition. The following code defines an **Event. OnOpenPage** function:

```
function Event.OnOpenPage(Page, Source)
-- does nothing !!
end
```

If a script defines this function, it will be called automatically by the filter before each page of the text-file is processed. The **`Page`** parameter passed to the function is a reference to the Sheet.Page table. The **`Source`** parameter passed to the function is a reference to the Column.Source table.

A definition of this function may include statements that will conditionally select the lines of text to be processed as columns.

```
function Event.OnOpenPage(Page, Source)
if  (strsub(Page.texts[5], 1, 8) ~= "Library:") then
    Source.fromline = 10 -- This is not a "header" page
    end
end
```

Because L4X saves the values assigned to fields **`Source.fromline`** and **`Source.toline`** and automatically restores these saved values before each call of this event function, the definition of the function does not need to re-assign the saved values to these fields.

2.3.2 Event.OnOpenRow

This field of the Event table may contain a function definition. The following code defines an **`Event.OnOpenRow`** function:

```
function Event.OnOpenRow(Page, Source)
-- does nothing !!
end
```

If a script defines this function, it will be called automatically by the filter before each line of the text-file is processed. The **`Page`** parameter passed to the function is a reference to thee Sheet.Page table. The **`Source`** parameter passed to the function is a reference to the Column.Source table.

A definition of this function may include statements that will conditionally exclude or condition lines of text.

```
function Event.OnOpenRow(Page, Source)
if  (strsub(Page.text, 81, 85) == "=====") then
     -- exclude this (and the following) line.
     Source.condition = -2
    end
end
```

Or, where the following columns are defined:

```
Column.Heading.A = {pos=1,  len=30,        condition=1}
Column.B         = {pos=40, fmt="number"}
Column.Total.C   = {pos=60, fmt="number", condition=2}
```

Then the following function definition, tests for a line beginning with a blank and, if true, selects the contents of **`Column.Heading.A`** only, but then skips to the next line without writing a new worksheet row:

```
function Event.OnOpenRow(Page, Source)
if  (strsub(Page.text, 1, 1) == " ") then
     -- selects contents of Column.Heading.A only.
     Source.condition = 1
    end
end
```

And the following function definition, tests for a line beginning with a non-blank character and, if true, selects the contents of **`Column.Heading.A`** and **`Column.B`**, and then adds a new row to the worksheet:

```
function Event.OnOpenRow(Page, Source)
if  (strsub(Page.text, 1, 1) ~= " ") then
     -- selects Column.Heading.A and Column.B
     Source.conditions[1] = 1
    end
end
```

Because L4X re-assigns zero values to field **`Source.condition`** and all elements of field **`Source.conditions[]`** before each call of this event function, the definition of the function does not need to re-assign zero to any of these fields or elements.

2.3.3 Event.OnWriteX

These fields of the Event table may contain a function definition. The following code defines an **`Event.OnWrite`** function:

```
function Event.OnWriteA(Page, Source)
-- does nothing !!
end
```

If a script defines this function, it will be called automatically by the filter before text is written to any cell in column: **A** of a worksheet. A script may contain a definition of an **OnWrite** function for any column or cell. The **`Page`** parameter passed to the function is a reference to table Sheet.Page. The **`Source`** parameter passed to the function is a reference to table Column.Source.

A definition of this function may include statements that return a value which will be written to the current cell.

```
function Event.OnWriteB(Page, Source)
     if  (strfind(Page.celltext, "TOTAL FOR") == 1) then
          return strsub(Page.text, 15, 66)
     end
end
```

The field: **`Page.celltext`** contains the text or number that will be written to the next cell of column: **B**. In this example, the script tests the value of this field, and if it contains **"TOTAL FOR"**, the function returns an alternative value, otherwise the contents of Sheet.Page.celltext will be written.

2.4 Sheet

This table contains fields that control how the worksheet is output. Each field of this table is described in detail in following sections of this document.

Fields of table: Sheet		
Name	*Type*	*Description*
Out	table	Contains a table that specifies miscellaneous parameters for worksheet output.
Page	table	Contains a table that describes the current state of the filter.
Pictures	table	Contains a table that specifies the conversion of text-strings into numbers or dates.
Write	function (deprecated)	Contains a pre-defined function. Use of this function is deprecated but is permitted for backwards compatibility. Please specify a value for field Sheet.Out.row instead.

2.4.1 Sheet.Out

This field of the Sheet is itself a table containing fields which specify values for miscellaneous properties of the worksheet and workbook output by the L4X filter.

The value assigned to field: **`Sheet.Out.row`**, which which specifies the worksheet row where fields **`A`** ... **`IV`** of the Column table will start writing, must be considered when formatting cells and columns. When this value is changed, a script may also need to adjust formatting and formulae in the "scripted" worksheet.

Fields of table: Sheet.Out		
Name	*Type*	*Description*
row	positive integer (optional)	Specifies the worksheet row in which column import starts. The formats and formulae of all the cells in this row are automatically copied and adjusted before each new row is imported. Default value is 1.
copycells	positive integer (optional)	By default L4X will copy all the cells in the start-row (see field: **`row`** above) and adjust any formulae. A script may specify a value to control how many cells in this row are copied. Default value is 0, all cells in the start-row are copied.
protect	boolean (optional)	Set this value to 0 if a script should un-protect output worksheet, or 1 if it is to be protected. Only un-protected worksheets in the "scripted" workbook can be un-protected worksheets in the output workbook. Default value is 1, protects the output worksheet.
sheetname	string (optional)	Specifies a new name for the worksheet containing this script.
vbsave	boolean (optional)	Set this value to 1 if a script should save the VB macros in the "scripted" workbook to the output workbook, or 0 to discard the VB macros. Default value is 0, VB macros are not saved.
bookpass	string (optional)	Specifies a password for the output workbook. If specified this password must be entered before the work book will open.

2.4.2 Sheet.Page

This field of the Sheet table is itself a table containing fields which describe the current state of the filter. A script should never assign values to this table or any of its fields. It is intended for use in event functions. when the script may need to test the contents of the fields in this table.

Fields of table: Sheet.Page		
Name	*Type*	*Description*
celltext	string (readonly)	Value is the string which will be written to a cell.
linecount	number (readonly)	Value is the number of lines in the page in process.
lineno	number (readonly)	Value is the line number of the line in process.
pagecount	number (readonly)	Value is the number of pages in the text-file.
pageno	number (readonly)	Value is the page number of the page in process.
poscount	number (readonly)	Value is the number of characters in each line of the text-file.
text	string (readonly)	Value is the text from the line in process.
texts[]	array of strings	Value is an array of text lines from the page in process. Each line of text can be refenced by the index operator, for example: texts[1] refers to the first line of text in the current page.

A script may conditionally define a column based on the value of a field in the **Sheet.Page** table, but as columns may **not** be defined in event functions, a script must use the **full** name of the Sheet. Page table. For instance:

```
if  (Sheet.Page.poscount >= 146) then
     Column.M = {pos=137, len=10, fmt="number"}
    end
```

2.4.3 Sheet.Pictures

This field of the Sheet table is itself a table containing fields which specify how text is converted to numbers.

The tables assigned to fields **A1**…**IV65535** of the Cell table and fields **A**…**IV** of the Column table may specify a value for the `fmt` field and each permitted value has a corresponding field in this `Sheet.Pictures` table. The value of this corresponding field is a "picture" string that controls how text is converted before it is written to a worksheet cell. If a script requires that a different picture-string be used for conversion from a particular format, it may assign a string to the corresponding field of this table.

Fields of table: Sheet.Pictures		
Name	*Type*	*Description*
number	string	Specifies a string that describes how text is to be converted to a number. Used when fmt="number" is specified. The default value "9.9-" defines the localized decimal point (within the nines) and a trailing sign character.
date	string	Specifies a string that describes how text is to be converted to a numeric date. Used when fmt="date" is specified. The default value "dd/mm/yy" defines the position of the two digit day dd, month mm and year yy within the text-string.
time	string	Specifies a string that describes how text is to be converted to a numeric time. Used when fmt="time" is specified. The default value "hh:nn:ss" defines the position of the two digit hour nn, minute nn and seconds ss within the text-string.
datetime	string	Specifies a string that describes how text is to be converted to a numeric date and time.Used when fmt="datetime" is specified. The default value is "dd/mm/yy hh:nn:ss".
boolean	string	Specifies a string that describes how text is to be converted to a worksheet boolean value. Used when fmt="boolean" is specified. The default value "yes" defines the (case-insensitive) text-string equivalent to true.
altnumber	string	Specifies a string that describes how text is to be converted to a number. Used when fmt="altnumber" is specified. The default value "(9.9)" defines the localized decimal point (within the nines) and leading/trailing sign characters.
altdate	string	Specifies a string that describes how text is to be converted to a numeric date. Used when fmt="altdate" is specified. The default value "dd/mmm/yy" defines the position of the two digit day dd, three-character month mmm and two-digit year yy within the text-string.
alttime	string	Specifies a string that describes how text is to be converted to a numeric time. Used when fmt="alttime" is specified.

		The default value "hh:nn" defines the position of the two digit hour nn and minute nn within the text-string.
altdatetime	string	Specifies a string that describes how text is to be converted to a numeric date and time.Used when fmt="altdatetime" is specified. The default value is "dd/mmm/yy hh:nn".
altboolean	string	Specifies a string that describes how text is to be converted to a worksheet boolean value. Used when fmt="altboolean" is specified. The default value "1" defines the text-string equivalent to true.
shortmonthnames	string	Specifies a string that describes short month names. These names are used when a date conversion picture uses mmm to specify the the month positions. The default value "JAN,FEB,MAR,APR,MAY,JUN, JUL,AUG,SEP,OCT,NOV,DEC" may be altered to suit other languages.
longmonthnames	string	Specifies a string that describes the long month names. These names are used when a date conversion picture uses mmmm (more than three m characters) to specify the the month positions. The default value "January,February, etc" may be altered to suit other languages.
delimit	string	Specifies the character(s) that are used to mark the end of each delimited-value. A script may use "*tab" to indicate that the delimiter is the tab character. The default value is ,
quote	string	Specifies the character that is used to "quote" a delimited-value. A delimited-value may be quoted when it could itself contain a delimiter character. If a quote character appears within the string it must be "escaped" by a preceding quote character. The default value is "

The following example shows how to assign a new picture string for cells and columns where **fmt="number"** is specified and the text to be converted to a number has a leading minus-sign and its decimal point is indicated by a comma.

```
Sheet.Pictures.number = "-9,9"
```

2.4.4 Sheet.Write

Use of this function is **deprecated** but is permitted for backwards compatibility. Please specify a value for field Sheet.Out.row instead.

This field of the Sheet table is a pre-defined function which reads the input text-file and imports selected text-strings into the worksheet. As the text-file is read, L4X processes cell and column tables and will automatically call any event functions functions defined in the script.

```
Sheet.Write([number | cell-reference | table])
```

By default, column import starts at the row **1** column **A** of the worksheet. However, the default start-row may be changed by passing an optional parameter, which may be one of:

- The number of the worksheet row where column import starts.
  `Sheet.Write(3)`

- The location of the worksheet cell where column import starts.
  `Sheet.Write("A4")`

- A reference to a table constructed like table **`Sheet.Out`**.
  `Sheet.Write({row=5})`

If used, the **Sheet.Write()** function call **must** be the **last** statement in a script.