

# 1 Contents



## The L4X Reference Manual (v1.0)

Intelligent text import for Microsoft Excel.

---

- [A Summary of the L4X Language](#)
  - [An Introduction to L4X Scripts](#)
  - [Selecting Input Text Strings](#)
  - [Output Locations for Selected Text](#)
  - [Output Locations for Repeated Text](#)
  - [Including or Excluding Lines of Text](#)
  - [Formatting Cell Contents](#)
  - [Filtering Input to Output](#)
  - [Event Functions](#)
  - [L4X Tables](#)
  - [Lua Tables and Types](#)
  - [Lua Functions](#)
  - [Lua Operators](#)
  - [Lua Control Structures](#)
  - [Cell.A1 .. IV65565](#)
  - [Column](#)
    - [Column.A .. IV](#)
    - [Column.Heading.A .. IV](#)
    - [Column.Source](#)
    - [Column.Total.A .. IV](#)
  - [Event](#)
    - [Event.OnOpenPage](#)
    - [Event.OnOpenRow](#)
    - [Event.OnWrite](#)
  - [Sheet](#)
    - [Sheet.Out,](#)
    - [Sheet.Page,](#)
    - [Sheet.Pictures,](#)
    - [Sheet.Write\(\)](#)
- [Supporting Material](#)

## 2 A Summary of the L4X Language

An L4X script controls how the L4X filter imports text into a worksheet. The L4X language extends [Lua 4.0](#) to include [tables](#) and [functions](#) that define how text is selected, converted and output.

L4X does **not** control the appearance and formatting of the worksheet it creates, it only specifies the cell contents of the worksheet. However, the resulting workbook is a copy of the original contents of the "example" workbook containing the L4X script that was run. L4X will output new columns of text to the copy of the worksheet that contained this L4X script and will automatically adjust any cell references in formulae and ranges which refer to locations on or below the row where columns were output. The position of all cells, pictures, and notes below this output row will also be adjusted automatically.

---

### An Introduction to L4X Scripts

The simplest possible L4X script directs the filter to read all (non-blank) lines from the input text-file and to [write](#) them to successive cells of column A in the output worksheet:

```
-- a comment line begins with two dashes --
Sheet.Write()
```

Notice that a script specifies neither the name of the input text-file nor the name of the output workbook/worksheet. These input and output files are opened by the environment in which L4X runs, before the script is interpreted:

L4X scripts can perform more complex imports. The following script selects the first **10** text characters, from all lines of text in the input file, and outputs them to successive cells of column A, beginning at row **2** of the output worksheet:

```
-- begin each script with a comment line
Column.A = {pos=1,len=10}
Sheet.Write(2)
```

The second statement in the above script defines `Column.A`, on the left-hand side of the [assignment](#) = operator, as an [output location](#) in the worksheet. A table, which defines how text-strings are [selected](#), is [constructed](#) on the right-hand side of the [assignment operator](#) = and assigned to `Column.A`.

You can specify that the text-strings output to column(s) are selected only from a range of lines within each selected page of text. The following script selects the first **20** text characters, from lines of text beginning at line **10** in each page of the input file, and outputs them to successive cells of column B, beginning at row **3** of the output worksheet:

```
-- another script
Column.Source = {fromline=10, toline=-1, frompage=1, topage=-1}
Column.B = {pos=1,len=20}
Sheet.Write(3)
```

Here, the values assigned to these fields of the [Column.Source](#) table specify a range of text lines. Only lines of text within this range will be processed by any [Column](#) tables that are defined in this script. 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. The other fields of the [Column.Source](#) table are described in a following section of this document, which describes how to [include or exclude lines of text](#)

For the sake of completeness, we provide the following L4X script which writes "hello, world" to the output-device:

```
--
```

```
print("hello, world")
```

The rest of this document explains how to write L4X scripts for the particular text imports that you require.

---

## Selecting Input Text Strings.

An L4X script may specify which particular text-strings are selected from a text file. To do this, a script must construct one Lua [table](#) for each string, or column of strings, to be selected. The following expression constructs and initializes a new table which defines the location and format of a single string:

```
{ page=2, line=10, pos=1, len=15, fmt="number" }
```

Here, the above expression constructs a table and, within the [table constructor](#) `{ }` operator, it assigns initial values to fields that describe the text-string to be selected. This table selects a text string, **15** characters long, from position **1**, on line **10** of the text in page **2** of the input text file. This selected string will be [converted to a number](#) before output to a worksheet cell.

If you want to select a series of strings from successive lines of text, then you should specify values for neither the line nor the page field. For instance:

```
{ pos=1, len=30 }
```

Here, the expression constructs and initializes a table which selects a series of text strings, **30** characters long, from position **1**, in all selected lines and pages. This selected string will **not** be [converted](#) before output to a worksheet cell.

Constructing a table does not, of itself, define an output location for the selected text, once constructed the table must be **assigned** to a named field of one the special [tables built into L4X](#). This process is discussed in the next section of this document.

---

## Output Locations for Selected Text.

An L4X script must specify the worksheet location(s) to which any selected text-strings are output. This is done by **assigning** the reference of each [text-selection](#) table it constructs to a named field of one the special [tables built into L4X](#). The name of this new field specifies the output location in a worksheet. For example, to write one text-string to one worksheet location, your script should include a statement like the following:

```
Cell.A1 = {page=2, line=10, pos=1, len=15, fmt="number"}
```

In the statement above, you can see, on the left-hand side of the [assignment](#) `=` operator, the field name **A1** of the "built-in" table [Cell](#). This field name specifies column **A**, row **1** as the worksheet location where a text-string will be output. On the right-hand side of the assignment operator you can see the [table constructor](#) `{ }` operator and the initial values for the fields within this table. A reference to this newly constructed table is then assigned to field **Cell.A1**.

If you want to output a series of text-strings selected from the input text file to multiple locations, you must construct a new table and assign its reference to a new field of table: [Column](#), as follows:

```
Column.B = {pos=1, len=30}
```

Here, the field name **B** in the [built-in](#) table [Column](#) specifies column **B** as the output location for selected text-strings. The row-number passed as a parameter of the [Sheet.Write\(\)](#) function determines the location of the first output cell within a column. After a text-string is output to a cell within a column, the output row number is automatically incremented.

---

## Output Locations for Repeated Text.

Cells in a worksheet row can contain data from both the detail and heading lines in the text file. If you want L4X to select a text string once per page but output this selected text repeatedly to **succeeding** cells of a worksheet, whenever a new row of cells is added by column processing, you may assign a new table to the appropriate field of the [Column.Heading](#) table. A script could include a statement like the following:

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

Here, the text string output to column **B** is selected from line **5** of every page, provided that line 5 is **not** within the lines specified by `Column.Source.fromline` and `Column.Source.toline`. This selected text string is then output to successive cells of column **B**, when Column **A** is output and a new row is added to the worksheet.

If you want L4X to select a text string from a line within the range of lines specified for **column** processing, but output this newly selected text string repeatedly to **succeeding** cells of a worksheet, whenever a new row of cells is added by column processing, a script could include a statement like the following:

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

Here, the text string output to column **B** is selected from position **10** of a line within the lines selected for column processing, provided that the value of field `condition` in table `Column.Source` is **1**. This newly selected text string is repeatedly output to successive cells of column **B**, whenever Column **A** is output and a new row is added to the worksheet. Please see the [Event functions](#) section of this document for instructions on changing the value of the `Column.Source.condition` field.

If you want L4X to select a text string selected from one of the lines specified for column processing, but output this selected text repeatedly to the **preceeding** cells of a worksheet, you may assign a new table to the appropriate field of the [Column.Total](#) table. A script could include a statement like the following:

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

Here, the text string output to column **C** is selected from position **99** of a line within the lines selected for column processing, provided that the value of field `condition` in table `Column.Source` is **2**. This newly selected text string is then output to any preceeding cells of column **C** which have not yet been written. Please see the [Event functions](#) section of this document for instructions on changing the value of the `Column.Source.condition` field.

## Including or Excluding Lines of Text.

The "built-in" table [Column.Source](#) selects pages within the input text-file and lines within each page. By default, only non-blank lines of text within this range will be selected for conversion by any of the [built-in](#) `Column` tables. You may also specify that, when a blank line is detected, processing for either the current page or the input text file is complete.

```
-- 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 table `Column.Source`, you do not need to construct and assign a new table. You need only assign a new value to this particular field. Once a table has been constructed and assigned you can refer to one of its fields using the `.` notation.

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

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

Here, the number assigned to field `linesperrow` specifies that four lines of text will be processed before a single worksheet row is output. If you want to select strings from the second or subsequent lines of text, you must assign an appropriate value to field `lineofrow` for each such column, 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
```

Expressions executed **within** the [Event function](#) `Event.OnOpenRow( )` can skip any number of lines on the current page by assigning a negative number to field `condition` of table `Column.Source`, like so:

```
-- exclude the current line
Source.condition = -1
-- exclude the current line and next line
Source.condition = -2
-- exclude this and all remaining lines on the current page
Source.condition = -9999
```

Please note that when you assign a **non-zero** number to field `Column.Source.condition`, this has the side effect of temporarily disabling all column processing. That is, output is not performed to any of the locations specified by field names **A - IV** of the [built-in](#) table [Column](#), nor is a row added to the worksheet.

## Formatting Cells Contents.

If your script outputs to cells which are formatted as number or date, text selected for the cell should be converted by L4X into the same format. Tables that describe how text-strings are selected may specify a value for the `fmt` field of the table, which describes the conversion required. Note that the format of all cells in the worksheet row where column output begins is automatically copied by L4X as each new row is inserted.

When L4X converts text-strings to numbers or dates, it uses a corresponding "picture" to control the conversion. The [built-in](#) 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, thousand, date and time separators and the order in which the sub-strings of date and time are specified. You may change these values 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, because you are only changing the value of **one** field in table `Sheet.Pictures` you do not construct and assign a new table, instead you just assign a new string value directly to the `date` field. Once a table

has been constructed and assigned you can refer to one of its fields using the . notation.

If only one column needs to be converted using a particular picture-string, then this may be specified in the table that defines the column's contents, like so:

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

L4X will always "protect" the output worksheet. You can allow the output worksheet's contents to be edited, if statement:

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

is executed before function [Sheet.Write\(\)](#) is called.

L4X will also remove any VBA macros in the output workbook. VBA macros can be included, if statement:

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

is executed before function [Sheet.Write\(\)](#) is called.

---

## Filtering Input to Output.

Once any required tables have been created and assigned, an L4X script **must** call the following function:

```
Sheet.Write(6)
```

Here the parameter value **6**, specifies that columns are output beginning at row 6 of the worksheet. This function **must** be called and should be the **last** statement of a script. Function [Sheet.Write\(\)](#) reads the input text-file and writes the output worksheet. While this function is executing it will call (at the appropriate points) any [event functions](#) that you may have defined in the script.

---

## Event functions.

Where the input text is not regularly formatted, your script may define [event functions](#). Event functions can inspect the current contents of the line and page of text which is being processed and control which columns are output. When a script defines an event function **before** the [Sheet.Write\(\)](#) function, the event function is called **automatically**, at each of the appropriate points in the conversion process. You may define a function for field [Event.OnOpenPage](#), which is called once before each page of text is processed. You may also define a function for field [Event.OnOpenRow](#), which is called once before each group of text-lines (selected for column processing) is processed.

The first function parameter passed in 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.linecount      -- count lines in page.
```

```

Page.lineno      -- line number of line in process.
Page.pagecount  -- count of pages in text file.
Page.pageno     -- page number of page in process.
Page.poscount   -- count of positions in text line.
Page.text       -- text from line in process.
Page.texts[]    -- array of text lines in page.

```

If a script needs to adjust the start line for column processing, you should define a function for field `Event.OnOpenPage`. For instance, when a text-file has "header pages" with fewer detail lines than succeeding pages, you might 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 your 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
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](#). Assigning a new value to field `Column.Source.fromline` has the side-effect of de-activating any fields assigned to the `Column.Heading` table. You may re-activate any or all of these `Column.Heading` fields, within the same event function, by assigning some value to the `Column.Heading.A.line` field. 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.

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

```

-- define your 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
end

```

Here, if the current line of text contains the characters "=====" at position 81 through 85, the script will set the value of field `condition` of table `Column.Source` to minus two. When the value of field `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 reset to zero before event functions are called, so a function does not need to set the zero condition explicitly.

Worksheet output defined by fields added to the [built-in](#) tables [Column.Heading](#) and [Column.Total](#) may be conditioned by the value of field `Column.Source.condition`. A script may change the value of this field in the function defined for field `Event.OnOpenRow`, as follows:

```

Column.Heading.A = {pos=1, len=30, condition=1}
-- define your event function

```

```

function Event.OnOpenRow(Page, Source)
if (strsub(Page.text, 1, 1) == " ") then
-- change condition - permits selection for column A.
Source.condition = 1
end
end

```

Here, the script changes the value of field `Column.Source.condition` when the current text line contains a blank character at position 1. If the value of this `Column.Source.condition` field is equal to the value of the condition field of table `Column.Heading.A`, then a new string is selected for column **A**. The value of field `Source.condition` is always reset to zero before this function is called, so this function does not need to set the zero value explicitly.

Assigning a positive number to field `Column.Source.condition` has the side effect of disabling all column processing. When you need to select both detail and heading/total strings from a line of text, you should assign positive numbers to one or more of the nine elements in the `Column.Source.conditions` array. For instance:

```

Column.Heading.A = {pos=1, len=30, condition=1}
Column.B         = {pos=40, fmt="number"}
Column.Total.C   = {pos=60, fmt="number", condition=2}
function Event.OnOpenRow(Page, Source)
if (strsub(Page.text, 01, 01) ~= " ") then
-- selects column A and Column B.
Source.conditions[1] = 1
end
if (strsub(Page.text, 60, 60) ~= " ") then
-- selects column C and Column B.
Source.conditions[2] = 2
end
end

```

Here, a new string is selected for column **B** and a new row is added to the worksheet, but the re-selection of text-strings for columns **A** and **C** is conditional. The value of all elements in the `Source.conditions` array are always reset to zero before this function is called, so this function does not need to set the zero value explicitly.

---

## L4X Tables

The L4X interpreter automatically builds the following [Lua tables](#) so that your scripts can describe the way in which input-text is selected, converted and output to a worksheet. The names of the fields which refer to these tables are reserved for use by L4X and should be used only as described below:

- [Cell](#)  
You 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 new table that describes how one text-string is extracted and converted before output.
- [Column](#)  
You 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 new table that describes how a series of text-strings are extracted and converted before output. Other [fields in the Column table](#), are described below.
- [Column.Heading](#)  
You 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 new table that describes how a series of text-strings are extracted and converted before being **repeatedly** output.



- [Column.Source](#)  
A reference to a table with fields that specify which pages and lines of text will be selected for **column** processing. You may replace the existing table with a new table or change the value of a one or more fields. For convenience, this table is passed as the **second** parameter of [Event](#) functions.
- [Column.Total](#)  
You 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 new table that describes how a series of text-strings are extracted and converted before being **repeatedly** output.
- [Event](#)  
A reference to a table with fields referring to functions that handle the events: [OnOpenPage](#) and [OnOpenRow](#).
- [Sheet](#)  
A reference to a table with fields that refer to the tables: [Out](#), [Page](#) and [Pictures](#) and the [Write\(\)](#) function that creates the output worksheet.
- [Sheet.Out](#)  
A reference to a table with fields that specify miscellaneous parameters values relating to worksheet output. You may replace the existing table with a new table or change the value of a one or more fields.
- [Sheet.Page](#)  
A reference to a table with fields that contain values from the text in the current line and page of the input file. For convenience, this table is passed as the **first** parameter of [Event](#) functions.
- [Sheet.Pictures](#)  
A reference to a table with fields that contain the "picture" strings which control how a text-string may be converted before output. You may replace the existing table with a new table or change the value of a one or more fields.

## Lua Tables and Types

Lua 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 **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
```

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"
```

A value of type **nil** is returned by any undefined variable name, or as a **false** result from the relational and logical operations described in a following section of this document.

---

## 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.

You may, of course, define your own functions in the script.

---

## 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** as false and a **none nil** value as true. Note the inequality operator is: ~=
  - Logical operators are: **and or not**  
Like Lua control structures, logical operators consider **nil** as false and anything else 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 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.
- 

## 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
```

Lua also provides **iterative** control structures **while** and **for**. Lua for PDFing scripts do not usually require iteration and these control structures should be used with caution.

## 2.1 Cell.A1 .. IV65535

You may specify the contents of any 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 possible cell location in a spreadsheet has a corresponding field name. A table constructed and assigned to this new field, itself specifies values for the following fields that determine how an input text-string is selected and converted.

Fields of table: Cell.A1 .. IV65535		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<b>pos</b>	<b>positive-integer</b> <b>(mandatory)</b>	Specifies the position of a text string in an input text line. If the text-string does not contain included blanks, you 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 you must specify the <b>left-most</b> position of the string and supply a value for the <b>len</b> field.
<b>len</b>	<b>positive-integer</b> <b>(optional)</b>	Specifies the length of a text string in an input text line. Used together with the <b>pos</b> field when the text-string contains included blanks.
<b>line</b>	<b>integer</b> <b>(mandatory)</b>	Specifies the line of the text-page from which the text-string is selected. You may specify a negative number where <b>-1</b> is the last line of a page, <b>-2</b> is the next to last line, and so on.
<b>toline</b>	<b>integer</b> <b>(optional)</b>	If you want a worksheet cell to contain data from more than one line of text, you must specify the last line of text from which the text-string is selected. You may specify a negative number where <b>-1</b> is the last line of a page, <b>-2</b> is the next to last line, and so on.
<b>page</b>	<b>integer</b> <b>(optional)</b>	Specifies the page of the text-file from which the text-string is selected. You may specify a negative number where <b>-1</b> is the last page of a file, <b>-2</b> is the next to last page, and so on. The default value is <b>1</b> , the first page of the file.
<b>fmt</b>	<b>string</b> <b>(optional)</b>	Specify <b>"number"</b> , <b>"date"</b> , <b>"time"</b> , <b>"datetime"</b> , or <b>"boolean"</b> when a text string must be converted to a numeric value. You may also specify the default value <b>"text"</b> when no conversion is required, or <b>"general"</b> , which provides automatic conversion to a numeric value when appropriate.
<b>picture</b>	<b>string</b> <b>(optional)</b>	If you specify a value for field <b>fmt</b> and the conversion <b>picture</b> in table <a href="#">Sheet.Pictures</a> is not correct for this particular field then you can specify the correct picture string here.

The following example will write the contents of a spreadsheet at location **A2** containing text selected from position **20**, line **4** and page **1** of the input text. The selected text will be converted to a number before output.

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

## 2.2 Column

This table contains fields that define how input text is selected and the worksheet columns to which the selected text is output.

Fields of table: Column		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<b>A .. IV</b>	<b>table</b>	You may add new fields to this <b>Column</b> table. The new field name must specify a spreadsheet column and its value must be a reference to a new <a href="#">table</a> that defines how text is selected for the named column.
<b>Heading</b>	<b>table</b>	Contains a reference to an <b>empty</b> <a href="#">built-in table</a> . You may add new fields to this table, the new field name must specify a worksheet column and its value must be a reference to a new <a href="#">table</a> that defines how text is selected and <b>repeatedly</b> output to succeeding cells in the named column.
<b>Source</b>	<b>table</b>	Contains a reference to a <a href="#">built-in table</a> that defines the range line of lines selected for column processing.
<b>Total</b>	<b>table</b>	Contains a reference to an <b>empty</b> <a href="#">built-in table</a> . You may add new fields to this table, the new field name must specify a worksheet column and its value must be a reference to a new <a href="#">table</a> that defines how text is selected and <b>repeatedly</b> output to preceeding cells in the named column.

Later sections of this document will describe each field of this table in detail.

## 2.2.1 Column.A .. IV

### Column.A .. IV

You may specify the contents of any worksheet column by constructing a table and assigning it to a new field of table **Column**. This new field may be named **A**, **B** and so on, up to **IV**, where every possible column location in a worksheet has a corresponding field name. A table constructed and assigned to this new field, itself specifies values for the following fields that determine how an input text-string is selected and converted.

When the input text is arranged in a column, you do not need to specify the location of every cell in the column. The text string specified for each Column will be extracted repeatedly from every line in the text-file selected for column processing and each text-string selected will be written to a new worksheet cell, new rows will be added to the worksheet as necessary.

Fields of table: Column.A .. IV		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<b>pos</b>	<b>positive-integer</b> <b>(mandatory)</b>	Specifies the position of a text string in an input text line. If the text-string does not contain included blanks, you 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 you must specify the <b>left-most</b> position of the string and supply a value for the <b>len</b> field.
<b>len</b>	<b>positive-integer</b> <b>(optional)</b>	Specifies the length of a text string in an input text line. Used together with the <b>pos</b> field when the text-string contains included blanks.
<b>lineofrow</b>	<b>positive-integer</b> <b>(optional)</b>	If the contents of one worksheet row is selected from multiple text-lines, you 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 <b>1</b> , the first text-line of the row.
<b>fmt</b>	<b>string</b> <b>(optional)</b>	Specify <b>"number"</b> , <b>"date"</b> , <b>"time"</b> , <b>"datetime"</b> , or <b>"boolean"</b> when a text string must be converted to a numeric value. You may also specify the default value <b>"text"</b> when no conversion is required, or <b>"general"</b> , which provides automatic conversion to a numeric value when appropriate.
<b>picture</b>	<b>string</b> <b>(optional)</b>	If you specify a value for field <b>fmt</b> and the conversion <b>picture</b> in table <a href="#">Sheet.Pictures</a> is not correct for this particular field then you can specify the correct picture string here.

The following example will add a column of cells at location **B** to the worksheet, selecting **30** characters of text (containing included blanks) from position **40** of each line of text selected for column processing.

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

Note that neither the line field nor page field is specified, because the range of lines selected for column processing is specified for all columns, by the table assigned to the [Column.Source](#) field.

## 2.2.2 ColumnHeading.A .. IV

You may specify the contents of any worksheet column by constructing a table and assigning it to a new field of table **ColumnHeading**. This new field may be named **A**, **B** and so on, up to **IV**, where every possible column location in a worksheet has a corresponding field name. A table constructed and assigned to this new field, itself specifies values for the following fields that determine how an input text-string is selected and converted.

Text-strings specified by fields of the **ColumnHeading** table will be selected, either when the value of the **line** field is outside the range of column processing or when the value of the **condition** field is equal to either the value of the **condition** field or the value of any element of the **conditions** array in the table referenced by the [Column.Source](#) field. Selected text 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 the detail and heading lines in the text file.

When either of the Event functions [OnOpenPage](#) or [OnOpenRow](#) are executed, a script can examine the current text page and set the value of fields in the table referenced by the [Column.Source](#) field, in order to select new text for this column.

Fields of table: ColumnHeading.A .. IV		
Name	Type	Description
<b>pos</b>	<b>positive-integer</b> <b>(mandatory)</b>	Specifies the position of a text string in an input text line. If the text-string does not contain included blanks, you 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 you must specify the <b>left-most</b> position of the string and supply a value for the <b>len</b> field.
<b>len</b>	<b>positive-integer</b> <b>(optional)</b>	Specifies the length of a text string in an input text line. Used together with the <b>pos</b> field when the text-string contains included blanks.
<b>line</b>	<b>integer</b> <b>(optional)</b>	Specifies the line of the text-page from which the text-string is selected. You may specify a negative number where <b>-1</b> is the last line of a page, <b>-2</b> is the next to last line, and so on.
<b>page</b>	<b>integer</b> <b>(optional)</b>	Specifies the page of the text-file from which the text-string is selected. You may specify a negative number where <b>-1</b> is the last page of a file, <b>-2</b> is the next to last page, and so on. The default value <b>0</b> , which selects the text on every page.
<b>condition</b>	<b>positive-integer</b> <b>(optional)</b>	Specifies the condition of the line. When a column is conditioned, text will be selected when either the value of field <b>condition</b> or the value of an element of array field <b>conditions</b> in table <a href="#">Column.Source</a> is set to the same value by an <a href="#">Event</a> function.
<b>fmt</b>	<b>string</b> <b>(optional)</b>	Specify <b>"number"</b> , <b>"date"</b> , <b>"time"</b> , <b>"datetime"</b> , or <b>"boolean"</b> when a text string must be converted to a numeric value. You may also specify the default value <b>"text"</b> when no conversion is required, or <b>"general"</b> , which provides automatic conversion to a numeric value when appropriate.
<b>picture</b>	<b>string</b> <b>(optional)</b>	If you specify a value for field <b>fmt</b> and the conversion <b>picture</b> in table <a href="#">Sheet.Pictures</a> is not correct for this particular field then you can specify the correct picture string here.

The following example will write the selected text to the next cell in the named column of the worksheet whenever a worksheet row is added.

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

You may specify the contents of any worksheet cell by constructing a table and assigning it to a new field of table **Column.Heading**. This new field may be named **A1**, **A2** and so on, up to **IV65535**, where every possible cell location in a worksheet has a corresponding field name. A table constructed and assigned to this new field, itself specifies values for the following fields that determine how an input text-string is selected and converted. The contents of the cell will be selected from position **20** of line **5**, when each new text-page is read, provided that the value of field [Column.Source.fromline](#) is greater than **5**.



### 2.2.3 Column.Source

The fields in this table determine how the input text file is processed. In particular, because a "text-selection" table assigned to a field of table **Column** specifies values for neither field **line** nor field **page**, the equivalent values are set for all fields in table **Column** by specifying values for fields in this **Column.Source** table.

The values of individual fields in this table may be set during the execution of [Event](#) functions.

Fields of table: Column.Source		
Name	Type	Description
<b>fromline</b>	<b>positive integer</b> (optional)	Specifies the starting text line of lines selected for column processing. Default value is <b>1</b> , the first line on the page.
<b>toline</b>	<b>integer</b> (optional)	Specifies the ending text line of lines selected for column processing. You may specify a negative number where <b>-1</b> is the last line of a page, <b>-2</b> is the next to last line, and so on. Default value is <b>-1</b> , the last line on the page.
<b>frompage</b>	<b>positive integer</b> (optional)	Specifies the starting text page of pages selected for column processing. Default value is <b>1</b> , the first page in the file.
<b>topage</b>	<b>integer</b> (optional)	Specifies the ending text page of pages selected for column processing. You may specify a negative number where <b>-1</b> is the last page of text, <b>-2</b> is the next to last page, and so on. Default value is <b>-1</b> , the last page in the file.
<b>linesperrow</b>	<b>positive-integer</b> (optional)	If the contents of one worksheet row is selected from multiple text-lines, you must specify the number of lines that will be used to compose each row. The default value is <b>1</b> .
<b>onblankline</b>	<b>string</b> (optional)	Specifies the action to be taken when a blank line of text is read. The string may be one of <b>"ignore"</b> , <b>"endpage"</b> or <b>"endtext"</b> . The default value is <b>"ignore"</b> .
<b>condition</b>	<b>integer</b> (optional)	Set in <a href="#">Event</a> functions to activate any fields in the <a href="#">Column.Heading</a> and <a href="#">Column.Total</a> tables with the same value specified for their <b>condition</b> field. Setting this value to a positive number will de-activate any "column" fields in the Column table. Setting this value to a negative number will cause the specified number of lines to be excluded from all processing. The value is set to zero, before event functions are called.
<b>conditions</b>	<b>array</b> (optional)	Set in <a href="#">Event</a> functions to activate any fields in the <a href="#">Column.Heading</a> and <a href="#">Column.Total</a> tables with the same value specified for their <b>condition</b> field. Any one of the nine elements of this array can be set, using the index operator [ ], to activate fields in the <b>Column.Heading</b> and <b>Column.Total</b> tables that have the same value specified for their <b>condition</b> field. These conditions are indexed by integers in the range <b>1</b> to <b>9</b> and are all 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}
```

You may set values for individual fields of this table in [Event](#) functions. The following code specifies values for elements of field `conditions`.

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

## 2.2.4 Column.Total.A .. IV

You may specify the contents of any worksheet column by constructing a table and assigning it to a new field of table **Column.Total**. This new field may be named **A**, **B** and so on, up to **IV**, where every possible column location in a worksheet has a corresponding field name. A table constructed and assigned to this new field, itself specifies values for the following fields that determine how an input text-string is selected and converted.

Text-strings specified by fields of the **Column.Total** table will be selected, whenever a text line is selected for column processing and the value of the **condition** field is equal to either of the values of the **condition** field or an element of the **conditions** array in the table referenced by the [Column.Source](#) field. Selected text 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 the detail and total lines in the text file.

When the Event function [OnOpenRow](#) is executing, a script can examine the current text line and set the value of fields in the table referenced by the [Column.Source](#) field, in order to select new text for this column.

Fields of table: Column.Total.A .. IV		
Name	Type	Description
<b>pos</b>	<b>positive-integer</b> <b>(mandatory)</b>	Specifies the position of a text string in an input text line. If the text-string does not contain included blanks, you 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 you must specify the <b>left-most</b> position of the string and supply a value for the <b>len</b> field.
<b>len</b>	<b>positive-integer</b> <b>(optional)</b>	Specifies the length of a text string in an input text line. Used together with the <b>pos</b> field when the text-string contains included blanks.
<b>condition</b>	<b>positive-integer</b> <b>(mandatory)</b>	Specifies the condition of the line. When a column is conditioned, text will be selected when either the value of field <b>condition</b> or the value of an element of array field <b>conditions</b> in table <a href="#">Column.Source</a> is set to the same value by an <a href="#">Event</a> function.
<b>fmt</b>	<b>string</b> <b>(optional)</b>	Specify <b>"number"</b> , <b>"date"</b> , <b>"time"</b> , <b>"datetime"</b> , or <b>"boolean"</b> when a text string must be converted to a numeric value. You may also specify the default value <b>"text"</b> when no conversion is required, or <b>"general"</b> , which provides automatic conversion to a numeric value when appropriate.
<b>picture</b>	<b>string</b> <b>(optional)</b>	If you specify a value for field <b>fmt</b> and the conversion <b>picture</b> in table <a href="#">Sheet.Pictures</a> is not correct for this particular field then you can specify the correct picture string here.

The following example, will write the selected text to all the preceding cells in the named column of the spreadsheet. If a preceding cell's already contains previously selected text, then this cell is not overwritten.

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

The contents of the cell will be selected from position **20** of the current line, provided that the value of field **Column.Source.Condition** is equal to **1**.

## 2.3 Event

The fields of this table contain the definitions of the **event** functions. If the input text file is not regularly formatted, you may define your own event functions to examine the text at run time and exclude particular lines of text and condition which columns are to be written.

Fields of table: Event		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<b>OnOpenPage</b>	<b>function</b> <b>(optional)</b>	A script may assign a <a href="#">function definition</a> to this field. If a script defines this function, it will be called automatically before each page of the text-file is processed.
<b>OnOpenRow</b>	<b>function</b> <b>(optional)</b>	A script may assign a <a href="#">function definition</a> to this field. If a script defines this function, it will be called automatically before each line of the text-file is processed.
<b>OnWrite</b>	<b>function</b> <b>(optional)</b>	A script may assign a <a href="#">function definition</a> to this field. If a script defines this function, it will be called automatically before each line of the text-file is processed.

Any Event functions that you define in your script are called automatically, when the [Sheet.Write](#) function is executed by your script. Your script should **never** call these functions explicitly.

### 2.3.1 Event.OnOpenPage

The following code defines the **Event.OnOpenPage** function:

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

|

If a script defines this function, it will be called automatically before each page of the text-file is processed. The **Page** parameter passed to the function is identical to table [Sheet.Page](#). The **Source** parameter passed to the function is identical to table [Column.Source](#).

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` when function [Sheet.Write](#) is called and automatically restores these saved values before each call of this event function, your definition of the function does not need to re-assign the saved values to these fields.

### 2.3.2 Event.OnOpenRow

The following code defines the **Event.OnOpenRow** function:

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

If a script defines this function, it will be called automatically before each line of the text-file is processed. The **Page** parameter passed to the function is identical to table [Sheet.Page](#). The **Source** parameter passed to the function is identical to table [Column.Source](#).

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
    Source.condition = -2 -- exclude this (and the following) line.
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
    Source.condition = 1 -- selects contents of Column.Heading.A only.
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 writes a new row to the worksheet:

```
function Event.OnOpenRow(Page, Source)
if (strsub(Page.text, 1, 1) ~= " ") then
    Source.conditions[1] = 1 -- selects Column.Heading.A and Column.B
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, your definition of the function does not need to re-assign zero to any of these fields or elements.

### 2.3.3 Event.OnWrite

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 whenever text is to be written to row: **A** of a worksheet. A script may contain a definition of an **OnWrite** function for any row or cell. The **Page** parameter passed to the function is identical to table [Sheet.Page](#). The **Source** parameter passed to the function is identical to table [Column.Source](#).

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

```
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 is field, and if it begins with "TOTAL FOR", the function returns an alternative value, otherwise the contents of **Page.celltext** will be written.

## 2.4 Sheet

This table contains fields that control how the worksheet is output. Later sections of this document will describe each field of this table in detail.

Fields of table: Sheet		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<b>Out</b>	<b>table</b>	Contains a reference to a <a href="#">built-in table</a> that defines miscellaneous parameters for worksheet output.
<b>Page</b>	<b>table</b>	Contains a reference to a <a href="#">built-in table</a> that contains the text currently being processed.
<b>Pictures</b>	<b>table</b>	Contains a reference to a <a href="#">built-in table</a> that controls conversion of text-strings into numbers or dates.
<b>Write</b>	<b>function</b>	Contains a reference to a pre-defined <a href="#">function</a> . A script <b>must</b> call this function to create a worksheet.



### 2.4.1 Sheet.Out

This table contains fields that specify miscellaneous parameters for worksheet output.

Fields of table: Sheet.Out		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<b>loc</b>	<b>string</b> <b>(optional)</b>	Specifies the output location for worksheet rows. This must be a valid cell reference, for example. <code>loc="A1"</code> . All column output locations will be adjusted so that their origin is relative to the location specified here.
<b>skipformulae</b>	<b>number</b> <b>(optional)</b>	Set to <b>1</b> , if you want to automatically skip formulae in the output row and adjust column output locations accordingly.
<b>copycells</b>	<b>positive integer</b> <b>(optional)</b>	By default L4X will copy <b>all</b> the cells in the output row and adjust any formulae. You may specify a value to determine how many cells in this row are copied.
<b>protect</b>	<b>boolean</b> <b>(optional)</b>	Set this value to <b>0</b> if you want the output worksheet to be unprotected, or <b>1</b> if it is to be protected. Default value is <b>1</b> , protects the output worksheet.
<b>vbsave</b>	<b>boolean</b> <b>(optional)</b>	Set this value to <b>1</b> if you want to save the VB macros in the example workbook to the output workbook, or <b>0</b> to discard the VB macros. Default value is <b>0</b> , VB macros are not saved.

## 2.4.2 Sheet.Page

This table contains information about the current process. You 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: <b>Sheet.Page</b>		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<b>celltext</b>	<b>string (readonly)</b>	Value is the string which will be written to the current cell.
<b>linecount</b>	<b>number (readonly)</b>	Value is the number of lines in the page in process.
<b>lineno</b>	<b>number (readonly)</b>	Value is the line number of the line in process.
<b>pagecount</b>	<b>number (readonly)</b>	Value is the number of pages in the text-file.
<b>pageno</b>	<b>number (readonly)</b>	Value is the page number of the page in process.
<b>poscount</b>	<b>number (readonly)</b>	Value is the number of positions in each line of the text-file.
<b>text</b>	<b>string (readonly)</b>	Value is the text from the line in process.
<b>texts[]</b>	<b>array of strings</b>	Value is an array of text lines from the page in process. Each line of text can be referenced by the index operator, for example: <b>texts[2]</b>

You may conditionally define a column based on the value of a field in the **Sheet.Page** table, before the `Sheet.Write ( )` function is executed. Because the condition is tested before any Event functions are called, you must reference the fields in 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

The tables assigned to fields of the [Cell](#) and [Column](#) tables may specify a value for the **fmt** field. Each permitted value for the **fmt** field has a corresponding field in this **Sheet.Pictures** table. The value of this field is a "picture" string that controls how text is converted before output to a worksheet cell. The **Sheet.Pictures** table has a default value for every permitted format. If you need to change the picture-string for a particular format you assign a string to the appropriate field of this table.

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

		The default value " <b>hh:nn</b> " defines the position of the two digit hour <b>nn</b> and minute <b>nn</b> within the text string.
<b>altdatetime</b>	<b>string</b>	Specify a string that describes how text is to be converted to a numeric date and time. Used when <b>fmt="altdatetime"</b> is specified. The default value is " <b>dd/mm/yy hh:nn</b> ".
<b>altboolean</b>	<b>string</b>	Specify a string that describes how text is to be converted to a worksheet boolean value. Used when <b>fmt="altboolean"</b> is specified. The default value " <b>1</b> " defines the text-string equivalent to <b>true</b> .
<b>shortmonthnames</b>	<b>string</b>	Specify a string that describes short month names. These names are used when a date conversion picture used <b>mmm</b> to specify the the month positions. The default value " <b>JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC</b> " may be altered to suit other languages.
<b>longmonthnames</b>	<b>string</b>	Specify a string that describes the long month names. These names are used when a date conversion picture used <b>mmmm</b> (more than three m characters) to specify the the month positions. The default value " <b>January,February, etc</b> " may be altered to suit other languages.

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

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

## 2.4.4 Sheet.Write

This **function** reads the input text-file and writes the output worksheet. As the text-file is read, L4X processes cell and column tables and will automatically call any [Event](#) functions defined in the script.

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

By default, column output originates at the row **1** column **A** of the worksheet. However, the default origin may be changed by passing an optional parameter, which may be one of:

- The number of the worksheet row where column output originates.  
`Sheet.Write(3)`
- The location of the worksheet cell where column output originates.  
`Sheet.Write("A4")`
- A reference to a table constructed like table [Sheet.Out](#).  
`Sheet.Write({loc="B5"})`

The L4X `sheet.write()` function **must** be coded and must be the **last** statement in a script.

### 3 Supporting Material for L4X

The L4X web site at: [www.pdfing.com/l4x](http://www.pdfing.com/l4x)

The L4X reference manual (PDF) at: [www.pdfing.com/l4x/down/l4xref.pdf](http://www.pdfing.com/l4x/down/l4xref.pdf)

The Lua 4.0 Language manual (PDF) at: [www.pdfing.com/l4x/down/luarefman.pdf](http://www.pdfing.com/l4x/down/luarefman.pdf)

The Lua home page at: [www.lua.org](http://www.lua.org)

The news-group: [gmane.comp.lang.lua.general](mailto:gmane.comp.lang.lua.general) hosted by: [news.gmane.org](http://news.gmane.org) and [Yahoo e-groups](http://Yahoo.e-groups)

---

#### Integrated Development Environment.

The integrated development environment for L4X (MS Excel for Windows only) can be downloaded from: <http://www.pdfing.com/l4x/down/l4x.msi>

---

#### Acknowledgements.

[Lua](#) is a copyright of [TegGraf, PUC-Rio](#).

The XlsReadWritell programming library, used to create Excel worksheets is by: [Axolot Data](#).

The L4X editor component is by: [Scintilla](#)

Excel is a trademark of: [Microsoft](#)

---

[This document](#) ©Jane Hearn 2007.  
All Microsoft trademarks acknowledged.

## 4 Change History for L4X V1.0

LuateX V1.1.0.3 - Released April 2005

- The table: [Sheet.Out](#) has an additional field: vbsave.

LuateX V1.1 - Released February 2005

- The table: Column.Range has been renamed to [Column.Source](#)  
Note that scripts referring to Column.Range will continue to work correctly.
- A reference to table: [Column.Source](#) is now passed as the second parameter to event functions.  
For convenience, in event functions, your scripts can use the shorter parameter name to refer to fields of this table.

L4X v1.0 - Released June 2007

---