LibreOffice RefCard

LibreOffice BASIC *Files*

v. 1.14 – 12/02/2018

Inter.

Written using LibreOffice v. 5.3.3 – Platform : All

Manipulating Files And Directories

Using BASIC Instructions

`Dir()`	Returns a file or directory name or the names of all files and directories found on a drive or directory which correspond to the specified path.
`FileCopy()`	Copies a file.
`FileDateTime()`	Returns a file date and time of a file creation or last modification, as a string.
`FileExists()`	Returns `True` if a file or directory exists.
`FileLen()`	Returns a file length (in bytes).
`GetAttr()`	Returns a file, volume or directory type.
`GetPathSeparator()`	Returns the path separation character for the operating system.
`Kill()`	Deletes a file from a drive.
`MkDir()`	Creates a new directory.
`Name()`	Renames an existing file or directory.
`RmDir()`	Deletes an existing directory.
`SetAttr()`	Defines a given file attributes.

Using A SimpleFileAccess Object

Methods of the `com.sun.star.ucb.SimpleFileAccess` service :

```
oSFA = createUNOService("com.sun.star.ucb.SimpleFileAccess")
```

`copy`	Copies a file.
`createFolder`	Creates a new directory
`exists`	Checks whether a file or directory exists.
`getContentType`	Returns a file contents type.
`getDateTimeModified`	Returns a file last modification date.
`getFolderContents`	Returns a directory contents
`getSize`	Returns a file size.
`isFolder`	Checks whether an URL is a directory.
`isReadOnly`	Checks whether a file is set as read-only.
`kill`	Deletes a file or directory. A directory is always deleted whether empty or not.
`move`	Moves a file.
`setInteractionHandler`	Declares an interaction handler for ulterior operations.
`setReadOnly`	Sets a file read-only flag (rights necessary).

File Paths

LibreOffice is multi-platform, thus file names are often expressed in URL form:
`file:///device/path/to/somefile.ext`

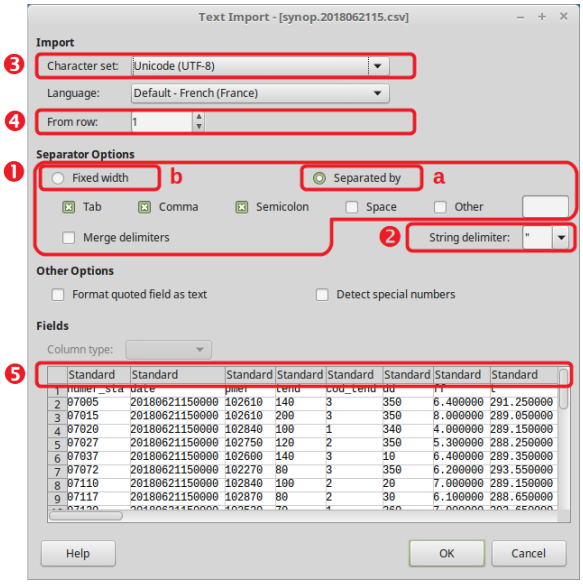
Conversion functions

From OS to URL	`UrlFileName = ConvertToURL(OSFileName)`
From URL to OS	`OSFileName = ConvertFromURL(URLFileName)`

Importing Text Files (CSV)

Using The GUI

Here's the Calc import dialog:



CSV Filters Items

CSV filters require 5 parameters (see the above figure for reference, #1 to #5). The final filter string is a wrap-up of all 5.

1. Field Separator

A. Variable Format

Separator ASCII code (see ASCII table)	`"9"`
Alternatives are separated with `/`	`"9/36"`
Merged characters: append `/MRG`	`"36/36/MRG"`

B. Fixed Format `"FIX"`

2. Text Delimiter

Delimiter ASCII code (see ASCII table) `"34"` (`""` if none)

3. Character Set

Frequently found character sets:

Windows-1252/WinLatin1	`"ANSI"`	ISO-8859-15/EURO	`"22"`
ISO-8859-1	`"12"`	UTF-8	`"76"` or `""`

Up-to-date list of supported sets: service `"com.sun.star.document.FilterFactory"`

4. First Line To Process

That line number (1-based) `"2"` (`"1"` or `""` if 1st line)

5. Column Formats

a. Variable	For each column, a 4-character sequence: `#col (1-based) / format /` (see the Column Formatting Codes table)	`"1/2/2/2/3/1"` (`""` when only default values)
b. Fixed	For each column, a 4-character sequence: `1st char pos. (0-based) / format /` (see the Column Formatting Codes table)	`"0/1/8/2/13/1"`

CSV Import Filter

The filter to use is created by concatenating the 5 parameters, using commas as separators:

① ② ③ ④ ⑤

```
Filtre = "9,34,76,1,1/2/2/2/3/1"
```

☞ Depending on the context, some values may be omitted (see parameters details) :
`Filter = "59,,76,,"`

More Information

Frequent ASCII Codes In CSV Files (Decimal Values)

9 Tabulation	34 "	36 $	44 ,	59 ;
32 Space	35 #	39 '	58 :	

Column Formatting Codes

1	Standard (automatic selection)	5	YY/MM/DD
2	Text	6	(ignore column)
3	MM/DD/YY	7	US Format (decimal and thousands sep.)
4	DD/MM/YY		

Importing A CSV File Into A Spreadsheet

Say we've got a CSV file named `MyFile.csv`. We want to copy its contents to `Sheet1` in the current document.
The copy is made by creating a link then deleting it:

```
Dim CsvURL As String        'the .csv source address
Dim Filter As String
Dim oSheet As Object        'the target sheet in the document

oSheet = ThisComponent.Sheets.getByName("Sheet1")
CsvURL = ConvertToURL("C:\path\to\MyFile.csv")
'csv file read options
Filter = "9,34,ANSI,1,1/2/2/2/3/1/4/1"
'import creating a link between the sheet and the .csv source
oSheet.link(CsvURL, "", "Text - txt - csv (StarCalc)", _
        Filter, com.sun.star.sheet.SheetLinkMode.VALUE)
'release link so that the document is independent
oSheet.setLinkMode(com.sun.star.sheet.SheetLinkMode.NONE)
```

☠ Any preexisting contents is **replaced** without warning.

Creating A Calc Spreadsheet From A CSV

We've got a CSV file named `MyFile.csv`. We want to create a new Calc document from this file contents.

```
Dim props1(1) As New com.sun.star.beans.PropertyValue
Dim props2()
Dim CsvURL As String        'the .csv source address
Dim DocURL As String        'the .odt target address
Dim oDoc As Object          'the target document

CsvURL = ConvertToURL("C:\path\to\MyFile.csv")
'csv file read options
props1(0).Name  = "FilterName"
props1(0).Value = "Text - txt - csv (StarCalc)"
props1(1).Name  = "FilterOptions"
props1(1).Value = "9,34,ANSI,1,1/2/2/2/3/1/4/1"
'load csv source file into the first sheet
oDoc = StarDesktop.loadComponentFromURL(CsvURL, "_blank", 0, props1())
'save to the ods target file
DocURL = ConvertToURL("C:\path\to\Spreadsheet.ods")
oDoc.storeAsURL(DocURL, props2())
```

☞ There's only one sheet in the newly created document. The sheet is named from the source file name.
☞ In the example, we display the created document. For hiding it:
– add a `Hidden` option with value `True` in `props1()`
– add `oDoc.close(True)` in the end.
☠ Any preexisting document with the same URL is **replaced** without warning.

Managing File Contents Using Instructions In BASIC

Process

1. **Get** an internal identifier on the file (`FreeFile`),
2. **Open** the file (`Open`),
3. **Write** to the file (`Print`, `Put` or `Write`) or **read** it (`Get`, `Line Input#` or `Input#`),
4. **Close** the file (`Close`)

Accessing A File Contents Using Its Handle (identifier).

`Close`	Closes a file that was opened using `Open`.
`Eof()`	Checks whether the file pointer is at the end.
`FileAttr()`	Returns the handle of a file that was opened using `Open`.
`FreeFile`	Returns an available handle number before opening a file.
`Get`	Reads a record from a file and inserts it into a variable.
`Input`	Sequentially reads an opened file data.
`Line Input`	Reads a line from a file.

`Loc()`	Returns the current position in an opened file.
`Lof()`	Returns an opened file size in bytes.
`Open`	Opens a data channel.
`Print`	Writes data into a sequential file (undelimited line).
`Put`	Writes a record into a file.
`Reset`	Close all opened files and flushes the contents write to the device of all file buffers.
`Seek()`	Returns the next read or write position in a file opened using `Open For Random` instruction.
`Write`	Writes data into a sequential file (delimited line).

☞ `Print` or `Write`?
`Print` saves the text as-is. `Write` adds delimiters according to the information type (text: ", date or boolean: #). These delimiters are ignored when read using `Input`.

Opening Modes (Text Files)

Sequential **write**: `Open For Output` Sequential **read**: `Open For Input`

☞ For other access modes (direct or binary access), use the stream APIs.

Sequential Read Example (Text File)

Read a file known from its `FileURL` address.

```
Dim Handle As Integer, TheLine As String
Handle = FreeFile
Open FileURL For Input As #Handle
'line reads
Do While Not Eof(Handle)
  'we read the current line
  Line Input #Handle, TheLine
Loop
Close #Handle
```

Sequential Write Example (Text File)

Write to a file known from its `FileURL` address.

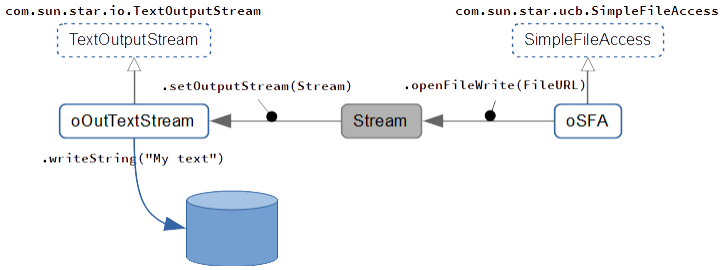
```
Dim Handle As Integer
Handle = FreeFile
Open FileURL For Output As #Handle
'line writes
Print #Handle, "Some text."
Print #Handle, "More..."
Print #Handle, "The end."
Close #Handle
```

Managing File Contents Using The Stream APIs

We call the LibreOffice API `SimpleFileAccess` and `Stream` services.

Principle

Example : writing to a text file (the actual code is shown in the example below).



Process

1. **Crate the object** for accessing files,
   `oSFA = createUNOService ("com.sun.star.ucb.SimpleFileAccess")`
2. **Connect** the stream that corresponds to the process (access type),
3. **Write** to the file or **read** it (according to the file type),
4. In write mode, **flush** the stream (`.flush`),
5. **Close** the file (`.closeXxx`).

Accessing File Contents

SimpleFileAccess Service (SFA)

`openFileRead`	Opens a file in read mode.
`openFileWrite`	Opens a file in write mode.
`openFileReadWrite`	Opens a file in write and read mode.

Stream Services (InputStream, OutputStream And Stream)

These are the “active” parts.
Correspondance between the SFA service and streams:

SFA methods	Stream services
`openFileRead`	`com.sun.star.io.InputStream`
	`com.sun.star.io.TextInputStream`
`openFileWrite`	`com.sun.star.io.OutputStream`
	`com.sun.star.io.TextOutputStream`
`openFileReadWrite`	`com.sun.star.io.Stream`

Stream Service

`getInputStream`	Returns the `InputStream` part of a read/write stream.
	☠ Closing this stream closes the `OutputStream` as well.
`getOutputStream`	Returns the `OutputStream` part of a read/write stream.
	☠ Closing this stream closes the `InputStream` as well.

InputStream Service

`readBytes`	Reads the specified number of bytes.
`readSomeBytes`	Reads the available number of bytes, with the maximum specified.
`skipBytes`	Skips the specified number of bytes (positive value).
`available`	Returns the number of bytes that can safely be read or skipped.
`closeInput`	Closes the stream.

TextInputStream Service

Inherits from `InputStream`.

`readLine`	Reads the text until a line break (`CR`, `LF`, or `CRLF`) or `EOF` is met. Returns the string read (without `CR` or `LF`). The characters read are converted according to the encoding set using `setEncoding`. If `EOF` was reached before running the method, a zero-length string is returned.
`readString`	Reads the text until a delimiter or `EOF` is met. Returns the string read. ☠ `CRLF` is **not** the default delimiter! This means that, when no delimiter is specified or found, the stream is read to `EOF`. The characters read are converted according to the encoding set using `setEncoding`. If `EOF` was reached before running the method, a zero-length string is returned.
`isEOF`	Returns the `EOF` status. ☠ This status cannot be detected by a read attempt for an empty string because such a response may be valid when using `readLine()` (the line is empty) or `readString()` (two delimiters inline).
`setEncoding`	Sets the character encoding (defaults to `UTF-8`). The available names are listed in this document: http://www.iana.org/assignments/character-sets (`Name` column). The current supported character sets depend upon the implementation.

OutputStream Service

`writeBytes`	Writes a sequence of bytes in the stream (blocking call).
`flush`	Flushes the stream buffers.
`closeOutput`	Used to signal that all data have been written.

TextOutputStream Service

Inherits from `OutputStream`.

`writeString`	Writes a string to the stream using the encoding specified using `setEncoding`. Line breaks and delimiters that might be necessary, have to be manually added to the string.
`setEncoding`	Sets the character encoding (defaults to `UTF-8`). The available names are listed in this document: http://www.iana.org/assignments/character-sets (`Name` column). The current supported character sets depend upon the implementation.

Example : Reading From A Text File

```
Dim oSFA As Object, oInText As Object
Dim FileURL As String, TheLine As String
oSFA = createUNOService("com.sun.star.ucb.SimpleFileAccess")
FileURL = ConvertToURL("C:\path\to\MyFile.txt")
oInText = createUNOService("com.sun.star.io.TextInputStream")
oInText.setInputStream(oSFA.openFileRead(FileURL))
TheLine = oInText.readLine()
oInText.closeInput()
```

Example : Writing To A Text File

```
Dim oSFA As Object, oOutText As Object
Dim FileURL As String
oSFA = createUNOService("com.sun.star.ucb.SimpleFileAccess")
FileURL = ConvertToURL("C:\path\to\MyFile.txt")
oOutText = createUNOService("com.sun.star.io.TextOutputStream")
oOutText.setOutputStream(oSFA.openFileWrite(FileURL))
'write (line delimiters must be specified)
'[here we use CRLF (Windows)]
oOutText.WriteString("Hello World" & Chr(13) & Chr(10))
oOutText.WriteString("Line #2" & Chr(13) & Chr(10))
'flush buffers ans close
oOutText.flush
oOutText.closeOutput()
```


Credits

Author : Jean-François Nifenecker – jean-francois.nifenecker@laposte.net

We are like dwarves perched on the shoulders of giants, and thus we are able to see more and farther than the latter. And this is not at all because of the acuteness of our sight or the stature of our body, but because we are carried aloft and elevated by the magnitude of the giants. (Bernard of Chartres [attr.])

History

Version	Date	Comments
1.12	20/06/2018	First EN version.
1.14	02/12/2018	Minor updates.