4/8/2018

Jagacy Software
www.jagacy.com

Jagacy Total Access

Total Access is a powerful data processing tool and report generator written in 100% Java -- it runs on any platform. It leverages the power of an enhanced awk scripting language that specializes in data conversion and management, cutting development time to a fraction of existing methods. Total Access can be used for data mining, generating adhoc analytics reports, and manipulating data from a variety of data sources. In addition, it is Web enabled, providing online report generation and viewing. It is an easy, fast, and cost-effective solution for your BI needs.

Table of Contents

1. Enhancements

Total Access is based on the awk programming language (please see https://www.gnu.org/software/gawk/manual/html_node/Getting-Started.html#Getting-Started, but disregard the gawk-specific items). It enhances the language by providing additional functions, and by allowing files other than text files to be processed. If the Input Type (IT) variable is set to "excel" or "csv", Total Access reads xls/xlsx or csv files, respectively. Each row is passed to the script as a line, with each column/field separated by the value of the Input Field Separator (IFS) variable. Please refer to the examples in the Total Access installation directory for more information.

2. Differences from awk

String literals	Supports \t, \n, \r, \b, \f, \", \', \\, \a, \v, \### (octal), \u#### (unicode), \x## (hexadecimal).
Number literals	Supports 0### (octal), 0x##, or 0x#### (hexadecimal).
Regular expressions	Of the form: https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html#sum
FS variable	If set to 1 space character (" "), matches on space (" "), \t, \n, \r, \f.
CONVFMT variable, printf(), sprintf()	Of the form: https://docs.oracle.com/javase/7/docs/api/java/util/Formatter.html#syntax
getline, getline < "file"	Subject to IT, IFS variable processing.

3. Variables

IT (Input Type) – Allows Total Access to read files other than text files. Acceptable values are:

- excel – The default sheet name is "Sheet1", booleans are not converted to 0 or 1, an empty row is returned as "null", and an empty column is returned as an empty string ("").
- excel:[sheet=<sheetName>,awkBoolean=<true|false>,emptyRow=<value>,emptyColumn=<value>]

- csv – The default delimiter is comma (“,”), an empty row is returned as “null”, and an empty column is returned as an empty string (“”).
- csv:[delimiter=<value>,emptyRow=<value>,emptyColumn=<value>]

- text (the default).

IFS (Input Field Separator) – Used to separate columns/fields in Excel and CSV input files. The default is tab (“\t”).

DATEFMT – The format used for dates. Of the form: https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html. The default is "yyyy-MM-dd HH:mm:ss.SSS".

SYSPROPS – The Java system properties as an associative array.

4. Array Functions

- **copyArray**(target, source)

  Copies the source array to the target array. Returns the target array.

- **createArray**(array, dimension, [subscript1, subscript2, …, subscriptN, value])

  Creates a sorted array. If dimension is 1, an integer with consecutive values are used as the subscript (of the form 1,2,3,4…), indexed from 1. Otherwise, the subscript argument(s) are used. Returns the array.

- **clearArray**(array)

  Rather than the delete statement, this function just clears the array for reuse. Returns the array.

- **mergeArray**(target, source1, [source2, source3, …])

  Merges source1, source2, etc. into the target array. Returns the target array.

- len = **arrayLength**(array)

  Returns the number of elements in the array.

- **subArray**(target, subscript, source)

Copies all source array elements that have the initial subscript to the target array. Returns the target array.

5. Data Type Functions

- type = **typeof**(var)

  Returns “AssocArray”, “Double”, “Integer”, or “String”.

- str = **string**(var)

  Returns the string value of var.

- dbl = **double**(var)

  Returns the double value of var, or 0 if it cannot be converted to a double.

- int = **integer**(var)

  Returns the integer value of var, or 0 if it cannot be converted to an integer.

6. Date Functions

- dateStr = **date**([format])

  Returns the current date/time as a string. If format isn’t specified, the value of the DATEFMT variable is used.

- dateStr = **convertDate**(fromDate, fromFormat, [toFormat])

  Converts the fromDate with the format fromFormat to toFormat. If toFormat isn’t specified, the value of the DATEFMT variable is used.

7. File Functions

- exists = **fileExists**(fileName)

  Returns 1 if the file exists, 0 otherwise.

- deleted = **deleteFile**(fileName)

  Deletes the file with the name fileName. Returns 1 if the file was deleted, 0 otherwise.

- renamed = **renameFile**(fromName, toName)

  Renames fromName to toName. Returns 1 if the file was renamed, 0 otherwise.

- created = **mkdir**(dirName)

  Makes the directory dirName. Returns 1 if the directory was created, 0 otherwise.

8. String Functions

- unescapedStr = **unescapeString**(escapedStr)

  Returns a string with all special characters (preceded by "\") unescaped.

- escapedStr = **escapeString**(unescapedStr)

  Returns a string with all special characters escaped (with "\").

- trimmedStr = **trim**(untrimmedStr)

  Returns a string with all beginning and ending whitespace removed.

9. Excel Functions

In addition to reading xls/xlsx files using the IT variable, Total Access provides the following functions. Please see the examples in the Total Access installation directory for more information:

- handle = **openExcel**(fileName, [sheetName])

  Opens a new or existing Excel spreadsheet, using the sheet sheetName. If the spreadsheet is new, the extension of fileName (.xls or .xlsx) determines what type of spreadsheet will be created. If sheetName is not specified, it defaults to "Sheet1". Returns a handle for all subsequent operations.

- **createExcelFont**(handle, fontId, fontStyles)

  Fonts are a limited resource, and should be reused. This function creates an Excel font with the name fontId, and applies the fontStyles to it (please see Excel Font Styles). The handle is the value returned by openExcel().

- **createExcelStyle**(handle, styleId, cellStyles)

  Cell styles are also a limited resource (3000), and should be reused. This function creates an Excel cell style and applies the cellStyles to it (please see Excel Cell Styles). The handle is the value returned by openExcel().

- **printExcelValue**(handle, row, column, value, [settings])

  Stores the value in the cell specified by row and column, applying the cell settings (please see Excel Cell Settings). Row and column are indexed from 1. The handle is the value returned by openExcel().

- **printExcelRTF**(handle, row, column, rtfStr, [settings])

  Stores the Rich Text Format (RTF) string in the cell specified by row and column, applying the cell settings (please see Excel Cell Settings). Row and column are indexed from 1. The handle is the value returned by openExcel().

- **printExcelDate**(handle, row, column, dateStr, dateFmt, [settings])

  Converts dateStr using dateFmt, and stores it in the cell specified by row and column, applying the cell settings (please see Excel Cell Settings). Row and column are indexed from 1. The handle is the value returned by openExcel().

- **printExcelFormula**(handle, row, column, formula, [settings])

  Stores the formula in the cell specified by row and column, applying the cell settings (please see Excel Cell Settings). Row and column are indexed from 1. The handle is the value returned by openExcel().

- **printExcelLink**(handle, row, column, value, link, [settings])

  Stores the hyperlink in the cell specified by row and column, applying the cell settings (please see Excel Cell Settings). Row and column are indexed from 1. The handle is the value returned by openExcel().

- **printExcelBoolean**(handle, row, column, value, [settings])

  Stores the boolean in the cell specified by row and column, applying the

cell settings (please see Excel Cell Settings). Row and column are indexed from 1. The handle is the value returned by openExcel().

- rowCount = **getExcelRowCount**(handle)

  Returns the number of rows in the sheet. The handle is the value returned by openExcel().

- columnCount = **getExcelColumnCount**(handle, row)

  Returns the number of columns in a row. Row is indexed from 1. The handle is the value returned by openExcel().

- value = **getExcelValue**(handle, row, column, [toAwkBoolean], [emptyRowValue], [emptyColumnValue])

  Returns the value of the cell at the row and column specified. If toAwkBoolean is 1, booleans are returned as 0 or 1. If emptyRowValue is set, this value will be returned for empty rows (otherwise "null" is returned). If emptyColumnValue is set, this value will be returned for empty cells (otherwise empty string ("") is returned). Row and column are indexed from 1. The handle is the value returned by openExcel().

- sheetName = **getExcelSheetName**(handle)

  Returns the sheet name being processed. The handle is the value returned by openExcel().

- line = **getExcelLine**(handle, outFields, [toAwkBoolean], [emptyRowValue], [emptyColumnValue])

  Returns the next row in an Excel sheet as a line. Cells in the line are separated by the value of the IFS variable. Cells are also stored in the outFields array. If toAwkBoolean is 1, booleans are returned as 0 or 1. If emptyRowValue is set, this value will be returned for empty rows (otherwise "null" is returned). If emptyColumnValue is set, this value will be returned for empty cells (otherwise empty string ("") is returned). The handle is the value returned by openExcel().

- **closeExcel**(handle, [closeSettings])

  Closes the Excel sheet, applying the closeSettings (please see Excel Close Settings).). The handle is the value returned by openExcel().

10. Excel Font Styles

Font Style	Possible Values
fontColor	See Excel Colors
bold	0 or 1
italic	0 or 1
strikeout	0 or 1
underline	none single double singleAccounting doubleAccounting
fontName	A valid font name.
fontPointSize	A valid point size.

11. Excel Cell Styles

Cell Style	Possible Values
useFont	A font id created with createExcelFont()
halign	center left right justify general
valign	center bottom top justify distributed
wrapText	0 or 1

dataFormat	Either a number:
	`0, "General"`
	`1, "0"`
	`2, "0.00"`
	`3, "#,##0"`
	`4, "#,##0.00"`
	`5, "$#,##0_);($#,##0)"`
	`6, "$#,##0_);[Red]($#,##0)"`
	`7, "$#,##0.00);($#,##0.00)"`
	`8, "$#,##0.00_);[Red]($#,##0.00)"`
	`9, "0%"`
	`0xa, "0.00%"`
	`0xb, "0.00E+00"`
	`0xc, "# ?/?"`
	`0xd, "# ??/??"`
	`0xe, "m/d/yy"`

	0xf, "d-mmm-yy" 0x10, "d-mmm" 0x11, "mmm-yy" 0x12, "h:mm AM/PM" 0x13, "h:mm:ss AM/PM" 0x14, "h:mm" 0x15, "h:mm:ss" 0x16, "m/d/yy h:mm" // 0x17 - 0x24 reserved for international and undocumented 0x25, "#,##0_);(#,##0)" 0x26, "#,##0_);[Red](#,##0)" 0x27, "#,##0.00_);(#,##0.00)" 0x28, "#,##0.00_);[Red](#,##0.00)" 0x29, "_(* #,##0_);_(* (#,##0);_(* \"-\"_);_(@_)" 0x2a, "_($* #,##0_);_($* (#,##0);_($* \"-\"_);_(@_)" 0x2b, "_(* #,##0.00_);_(* (#,##0.00);_(* \"-\"??_);_(@_)" 0x2c, "_($* #,##0.00_);_($* (#,##0.00);_($* \"-\"??_);_(@_)" 0x2d, "mm:ss" 0x2e, "[h]:mm:ss" 0x2f, "mm:ss.0" 0x30, "##0.0E+0" 0x31, "@" - This is text format. 0x31 "text" - Alias for "@" Or a custom string format (please see https://support.microsoft.com/en-us/help/264372/how-to-control-and-understand-settings-in-the-format-cells-dialog-box)
fillForegroundColor	(Note: Set foreground color before background color) See Excel Colors
fillBackgroundColor	(Note: Set foreground color before background color) See Excel Colors
fillPattern	altBars, bigSpots, bricks, diamonds, fineDots, leastDots, lessDots, noFill, solidForeground, sparseDots, thickBackwardDialog, thickForwardDialog, thinHorzBands, thinVertBands
borderBottom, borderLeft, borderRight, borderTop	dashDot, dashDotDot, dashed, dotted, double, hair, medium, mediumDashDot, mediumDashed, none, slantedDashDot, thick, thin
borderBottomColor, borderLeftColor,	See Excel Colors

borderRIghtColor, borderTopColor	
shrinkToFit	0 or 1
quotePrefix	0 or 1
locked	0 or 1
hidden	0 or 1
rotation	A valid rotation value.
indentation	A valid indentation value.

12. Excel Cell Settings

Cell Settings	**Possible Values**
useCellStyle	A style id created with createExcelStyle()
activeCell	1

13. Excel Close Settings

Close Settings	**Possible Values**
selectedSheet	1
activeSheet	1
hiddenSheet	1
autoSizeColumn, columnNumber	1. Note that columnNumber is indexed from 1.
columnWidth, columnNumber	width. Note that columnNumber is indexed from 1. width is 1/256 of a character.

14. Excel Colors

One of the following:

aqua, automatic, black, black1, blue, blueGrey, blue1, brightGreen, brightGreen1, brown, coral, cornflowerBlue, darkBlue, darkGreen, darkRed, darkTeal, darkYellow, gold, green, grey25%, grey40%, grey50%, grey80%, indigo, lavender, lemonChiffon, lightBlue, lightCornflowerBlue, lightGreen, lightOrange, lightTurquoise, lightTurquoise1, lightYellow, lime, maroon, oliveGreen, orange, orchid, paleBlue, pink, pink1, plum, red, red1, rose, royalBlue, seaGreen, skyBlue, tan, turquoise, turquoise1, violet, white, white1, yellow, yellow1.

15. CSV Functions

In addition to reading CSV files using the IT variable, Total Access provides the following functions:

- handle = **openCsv**(fileName, [delimiter])

  Opens an existing CSV file. If delimiter is not specified, it defaults to comma (“,”). Returns a handle for all subsequent operations.

- line = **getCsvLine**(handle, outFields, [toNumber], [emptyRowValue], [emptyColumnValue])

  Returns the next line in a CSV file. Fields in the line are separated by the value of the IFS variable. Fields are also stored in the outFields array. If toNumber is 1, Total Access attempts to convert the field into a number before storing it in the outFields array. If emptyRowValue is set, this value will be returned for empty rows (otherwise “null” is returned). If emptyColumnValue is set, this value will be returned for empty cells (otherwise empty string (“”) is returned). The handle is the value returned by openCsv().

- **closeCsv**(handle)

  Close a CSV file. The handle is the value returned by openCsv().

- line = **parseCsvLine**(input, outFields, [delimiter], [toNumber], [emptyRowValue], [emptyColumnValue])

  Parses the CSV input and returns a line. Fields in the line are separated by the value of the IFS variable. Fields are also stored in the outFields array. If delimiter is specified, it is used as the delimiter for the CSV input; otherwise comma (“,”) is used. If toNumber is 1, Total Access attempts to convert the field into a number before storing it in the outFields array. If emptyRowValue is set, this value will be returned for empty rows (otherwise “null” is returned). If emptyColumnValue is set, this value will be returned for empty cells (otherwise empty string (“”) is returned

- str = **unescapeCsv**(str)

  Returns an unescaped CSV string.

- str = **escapeCsv**(str)

  Returns an escaped CSV string.

16. Fixed Width Functions

Please see the examples in the Total Access installation directory:

- line = **parseFixedLine**(input, widths, outFields, [toNumber])

  Parses the fixed width input and returns a line. Fields in the line are separated by the value of the IFS variable. Fields are also stored in the outFields array. The widths array specifies the width of each field (and white space). If toNumber is 1, Total Access attempts to convert the field into a number before storing it in the outFields array.