10 Spreadsheet Integration

Version 10.20.00, 2013-03-30

TSTool and spreadsheet software offer many similar capabilities. However, there are significant differences in the software features and approaches to structuring and processing data. It may be appropriate to use one tool and not the other for certain tasks. And, it also may be appropriate to use the tools in conjunction (whether in sequence or in parallel). This chapter provides a comparison of TSTool and Microsoft Excel and describes how they can be integrated. Microsoft Excel is the spreadsheet of choice in this chapter; however, other spreadsheet software, such as Open Office Calc, are similar, in particular when **.xls*, **.xlsx*, and **.csv* file formats are used. Common TSTool and Excel integration needs include:

- How can Microsoft Excel be executed from TSTool?
- How can TSTool be executed from Excel?
- How can data in an Excel file be extracted (pulled) by TSTool?
- How can data in TSTool be inserted (pushed) into an Excel file?
- How can data produced by TSTool be imported (pulled) into an Excel file?
- How can data produced by Excel be exported (pushed) into a TSTool file?
- How can TSTool results (e.g., charts and formatted reports) be made to look "more like Excel" (this may be related to functionality or simply a cosmetic preference for Excel)

Integrating TSTool with Excel can be challenging because:

- TSTool software is written in Java and Excel uses Microsoft Technologies – the toolkits for integration are different and involve different knowledge and skills
- Excel file formats may be binary, often are complex, and formats may not have clear documentation (it often is necessary to reverse engineer Microsoft formats) and
- The complexity of Excel files make it difficult to simply manipulate the files and therefore libraries need to be used to retain the integrity of the files

TSTool and Excel can be run in sequence to perform data processing. This approach can involve manual or automated execution. Both approaches require appropriate integration, whether using intermediate data files or direct connections, and direct connections must ensure that sufficient error handling is in place.

Many software developers and users face the challenges of integrating their software with Excel and various solutions are available. The appropriateness of any approach depends on the specific need and technologies that are available. This remainder of this chapter explains how to integrate TSTool with Excel, and will be expanded as successful techniques are confirmed. The focus is on recent software versions and technologies.

10.1 TSTool and Excel Comparison

The following table compares features of TSTool and Excel. In summary:

- TSTool processes data using a sequential workflow of commands, whereas Excel processes data using an internal solver that understands data dependencies.

- TSTool provides immediate feedback on errors when editing commands, but does not fully process data until commands are run; Excel processes formulas immediately and provides feedback on errors.
- TSTool is designed to separate data and processing logic whereas Excel allows users to mix.
- TSTool data representations are generally human-readable, whereas Excel files are binary and can be difficult to interpret.

TSTool and Excel Comparison

Feature	TSTool	Excel
Software language	Java	Microsoft technologies, depend on version.
File extension	**.TSTool* for commands, others for data and configuration files.	**.xls* (pre-2007 version), binary **.xlsx* (2007 version) zipped XML.
Data file formats supported	See datastore appendices for time series formats, also DBF and CSV files for tables.	**.xls*, **.xlsx*, **.dbf*, and other formats.
In-memory data representation	Lists of objects: • Time series • Tables • Processor properties	Workbook of worksheets, each containing a grid of cells, with each cell having a type (e.g., number, text, formula) and formatting properties.
Data processing logic representation	Commands text, saved in text command file. Commands used named parameters (`Parameter="Value"`) that allow defaults and any parameter order. Standards are enforced or recommended by TSTool. For example, TSTool uses a standard time series identifier (TSID) convention.	Formulas defined for cells (select cell to see formula). Formula arguments must be in a specific order, although arguments are optional in some cases. Excel is a free-form tool and does not impose many standards. However, there are limitations such as character restrictions for named ranges.
Data processing mechanism	Commands access data objects and manipulates their contents.	A formula in a cell modifies one or more cells.
Command editing	Editor dialogs are provided for editing commands in TSTool. Text command files can be edited directly.	Formula editors are provided and can also be edited as the cell contents.
Data and processing separation	Processing logic and graph configuration is clearly separated from data.	Data can be saved on separate sheets, with computations on other sheets. However, it is easy to mix data and processing logic.
User-defined code	• Call external Python script or other software externally (see General commands) • Templates • Plug-in commands (planned for future)	• Macros programmed in VBA • Call external programs • Third party extensions • Microsoft libraries (e.g., available in IronPython)
Third-party plug-ins	Not yet supported in integrated way but can read/write file formats and call Python, etc.	Available from third parties.

Feature	TSTool	Excel
Processing workflow	Sequential, first command to last (subset of commands can be executed).	Based on cell formula dependencies. (Excel contents are kept up to date as per cell contents and user interactions).
Transparency	Commands and data are visible as text, can review sequential processing.	Formulas can be viewed, sequence of processing is determined by Excel based on cell dependencies and may not be obvious to user.
Data identifiers	Time series and tables have identifiers. Time series also have aliases. Time series identifiers are based on TSTool conventions and data from the original source.	Named ranges can be assigned and Excel allows tables to be defined using column headings.
Data size limitations	None, other than memory of computer and Java virtual machine.	Excel row and column limits vary by Excel version.
Scalability	Built into design of data management and processing, for example with TSList command parameters that accept wildcards, database query filters, and templates.	Add more worksheets, columns, and rows.
Templates	Used to scale processing of large amounts of data.	Not available (have to replicate worksheets, rows, columns).
Automation	Fundamental part of TSTool design.	Workbooks tend to be an accumulation of data and processing, although Excel can be called in a process to perform specific work.
Command line (batch mode)	Run with: `tstool -commands …`	Can run `excel.exe` with switches.
Time series graphs	• Built in with default properties • Extensive graph properties • Can automate graphs using text time series product files and `ProcessTSProduct()` command. • Focus on time series, not general data series • Can graph different data intervals on same chart	• Built in charts • Use third party tools • Time series can be graphed but are essentially arrays of numbers • Cannot easily graph different intervals on same chart due to grid formatting of data
Database connectivity	• Integrated for specific datastores • Use ODBC DSN defined for Excel workbook and GenericDatabaseDataStore	• Use ODBC DSN defined for Excel workbook
Accessing external data	Use read commands to read from standard file formats, databases, and web services, including delimited files.	Use features under the ***Data*** menu, including accessing from databases, web services, and files.
Missing data	Handled transparently in most cases, including special data values (e.g.,	No specific handling, typically must ensure blanks in data cells so that

Feature	TSTool	Excel
	`NaN, -999`).	missing values will not generate errors.
Data units	Handled transparently in most cases, based on units in time series metadata, with checks in place to prevent incompatible manipulation.	No specific handling. User must guard against incompatible manipulation.
Data validation	Commands are available to check and compare data.	Formulas can be used, can enable data validation for input controls.
Software/logic testing	Built in framework and commands available to automate testing.	Not sure.
Logging	Built in, with text log file.	Not sure.

10.2 Running TSTool from Excel

Reasons for running TSTool from Excel include:

- TSTool is used to acquire and/or process data as input to Excel
- TSTool performs an analysis/visualization function

To run TSTool from Excel, run it like any other external system call, using a VBA macro in Excel (see: "How to Launch a Win32 Application from Visual Basic http://support.microsoft.com/kb/129797). TSTool can be run in batch mode using a command line:

```
TSTool -commands CommandFile.TSTool
```

See the **Getting Started** chapter for more information about running TSTool in batch mode. In the future, a TSTool server mode may be implemented to allow Excel to "drive" TSTool, for example using REST web services.

10.3 Running Excel from TSTool

TSTool can run Excel by using its `RunProgram()` command, and appropriate Excel command-line switches (see: http://office.microsoft.com/en-us/excel-help/command-line-switches-for-excel-HA010158030.aspx) (is there a better reference, for example to explain how to run in headless mode and exit?).

It also is possible to run and control Excel. However, this is not integrated into TSTool. One option is to use the TSTool `RunPython()` or `RunProgram()` command to run an IronPython script, which then interacts with Excel. For example, see:

http://www.ironpython.info/index.php/Interacting_with_Excel

10.4 Manipulating Excel Files from TSTool

Excel files (**.xls*, **.xlsx*) contain the definition of a workbook, worksheets, data, formulas, formatting, charts, and other information. Excel files are complicated and can be difficult to manipulate, especially when trying to ensure that different versions of Excel files are properly handled. Although it may be possible to modify the files directly (e.g., search and replace strings in the **.xlsx* XML), this can lead to file corruption.

A more robust way to manipulate Excel files is to utilize a software package or library that has been written specifically to manipulate Excel files through an application programmer interface (API). The following options are available for integrating TSTool and Excel and an appropriate option should be chosen for the specific environment, problem, and user:

- Use built-in TSTool features to interact directly with Excel **.xls*, **.xlsx* files:
  - The TSTool `ReadTableFromExcel()` command will read a table from Excel and the table can be processed by other table commands. For example, use the `TableToTimeSeries()` command to convert data read from Excel into time series that can be further processed with TSTool commands.
  - There currently is no TSTool command implemented to write to an Excel workbook, although this may be implemented in the future.
  - There currently is no TSTool command implemented to read time series directly from an Excel workbook (without using the `TableToTimeSeries()` command), although this may be implemented in the future.
  - There currently is no TSTool command implemented to write time series directly to an Excel workbook, although this may be implemented in the future.
  - The Apache POI package (http://poi.apache.org) is used for integration with Excel files. Functionality that is envisioned for future TSTool enhancements can be implemented using this package.
- Use built-in TSTool features to interact directly with Excel **.csv* files, which can be edited by Excel and do not include formatting, formulas, etc.:
  - Use the `ReadDelimitedFile()` command to read time series from a delimited file that was exported from Excel as a **.csv* file.
  - Use the `ReadTableFromDelimitedFile()` command to read a table from a delimited file that was exported from Excel as a **.csv* file.
  - Use the `WriteDateValue()` command to write a DateValue format file. The bottom part of the file can be read into Excel as a delimited file. A command may be implemented in the future to write time series to a delimited file (with no extra information).
  - Use the `WriteTableToDelimitedFile()` command to write a TSTool table to a delimited (**.csv*) file that can be opened in Excel.
- Use built-in TSTool features to interact with an Excel file as if it is a database:
  - Define an ODBC DSN for the Excel file and use a GenericDatabaseDataStore (see **Generic Database DataStore** appendix). Use `DatabaseEngine="Excel"` and specify the `OdbcName` property in the datastore configuration file.
  - Use the `ReadTableFromDataStore()` command:
    - The Excel worksheet must be relatively simple with column headings in the first row.
    - Need a good reference for EXCEL SQL statements, in particular showing complex queries and how to write to the sheet? What are the limitations on SQL when processing Excel files?
  - Optionally use the `TableToTimeSeries()` command to convert the table to time series.
  - There currently is no command to read a datastore table into a time series (without using the `TableToTimeSeries()` command); however, this capability may be added in the future.
- Use Python software to read/write Excel files and create files that can be processed with TSTool commands, such as a **.csv* file and the `ReadTableFromDelimitedFile()` command. The

use of Python or similar adds another level of communication between TSTool and Excel, but may be necessary to achieve the level of data manipulation and integration that is required. Python versions that have been tested with TSTool's `RunPython()` command include:

- Python xlrd (http://pypi.python.org/pypi/xlrd) and xlwt modules (http://pypi.python.org/pypi/xlwt):
  - The xlwt module does not handle **.xlsx* files as of version 0.7.4
  - Requires understanding the internals of Excel data representations
- IronPython (see: http://www.ironpython.info/index.php/Interacting_with_Excel):
  - Uses native Microsoft .NET libraries (tighter integration) but IronPython lags behind Python
  - Requires understanding the internals of Excel data representations
- Jython (http://www.jython.org/):
  - Is written in Java (see the TSTool `RunPython()` command for the version that is shipped with the TSTool software installer)
  - Has potential to allow running Python scripts with tight integration to TSTool software, including access to TSTool internal objects

10.5 Manipulating TSTool Files from Excel

There may be a need to create TSTool files from Excel. For example, TSTool supports the DateValue file format, which is essentially a delimited file with additional header information with time series properties. Other TSTool time series data files are text files that adhere to specifications of data providers (e.g., model formats). TSTool time series product files that describe graphs and commands are documented and can be written out with Excel or other software. Excel macros can be written to create these files and then TSTool can be called from Excel as described above.

In the future, standard Excel macros may be distributed with TSTool to facilitate Excel import/export capabilities.

10.6 Testing Excel and TSTool Software and Command Files

The **Quality Control** chapter explains how to use TSTool features for testing, including testing commands and user-defined command files. It also is possible to use TSTool test features in conjunction with Excel:

- to understand the similarity or differences between TSTool commands and Excel formulas (differences may be by design, due to precision of analysis, etc.)
- to validate a TSTool command by comparing to Excel, in particular for more complex functionality that cannot easily be validated with visual inspection
- to compare an analysis performed with multiple TSTool commands and the corresponding Excel formulas (for example to prototype the analysis in one of the tools and then fully implement in the other)

Specific issues related to testing with Excel include:

- TSTool's limited ability to read Excel files may limit testing. For example, see the `ReadTableFromExcel()` command documentation for limitations. One known issue is that Excel formula results may not be accessible to the `ReadTableFromExcel()` command because the formula results in the cell may be computed by Excel when the workbook file is opened and is not stored in the file. In this case it may be necessary to copy the Excel worksheet,

paste special into a new worksheet, copying the values (or copy formatting first and then copy values). The values can then be read and used in the test. This is not ideal for reading Excel files for operational processing but may be a reasonable work-around for defining software tests.

- Some Excel integration approaches such as using a datastore or IronPython may result in Excel software (or underlying "headless" library processes) processes running that may impose restrictions such as locking files. If issues arise the work-around may be to read Excel files directly with `ReadTableFromExcel()` and similar commands.

10.7 Making TSTool Graph Output Look Like Excel Charts

TSTool has been developed primarily as a tool that automates and streamlines data processing. Features to visualize time series have been implemented in Java whereas Microsoft Excel is developed in Microsoft technologies. It has not been a focus to implement all TSTool visualization features to match Excel; however, it is recognized that many users are familiar with Excel charting and would like to produce similar charts in TSTool. The following table lists areas where TSTool does not match Excel, the rationale for the difference, and options for improvements that can be implemented in the future.

TSTool and Excel Time Series Graph Comparison

TSTool Graph Feature Compared to Excel	Rationale for Implementation in TSTool	Potential enhancement
TSTool default graphs are simple, with no titles	The default graphs that are displayed have relatively basic formatting because TSTool processes data from many sources. For example, it is not implemented in the current software to use blue to draw all time series of data type "streamflow". Time series properties for titles are also difficult to implement in generic way.	Allow default graph properties to be configured so that TSTool defaults are more appropriate for a particular system. Users can edit many graph properties by right-clicking on graph area and use the `ProcessTSProduct()` command.
TSTool has limited fonts	The core fonts supported in Java are offered. It can be an issue if non-standard fonts are used because they are not offered on all computers.	Expand fonts based on latest Java capabilities and use standard default if font is not available.
TSTool X-axis graph labels cannot be rotated	The label features were implemented with an older version of Java that did not support rotated text. Also, the TSTool auto-labeling features for date/times provide intelligent horizontal labeling.	Enable label rotation as an option using latest Java capabilities.
TSTool Y-axis graph labels cannot be rotated	See above. Vertical Y-axis labels would be beneficial, especially for long Y-axis labels. The current work-around is to show the Y-axis label horizontally at the top of the axis.	Enable label rotation as an option using latest Java capabilities.

TSTool Graph Feature Compared to Excel	Rationale for Implementation in TSTool	Potential enhancement
TSTool lines look a bit blocky	TSTool has no limitations on the number of data points that can be graphed. However, handling missing data as gaps in lines, sometimes results in blocky transitions through sharp angles.	Improve the TSTool drawing algorithm to take advantage of improved Java drawing internals (use less moveto/lineto and instead draw more polylines). Also implement a line property to draw lines by connecting points (the current default) and as step-function to show continuous value for interval.
TSTool bar charts do not look as nice as Excel	TSTool bar charts are designed to handle large number of data points and the drawing logic sometimes results in non-optimal space between bars.	Spend more time looking at bar representation, in particular case when bars are left off the ends of the graph.
TSTool legend default is too complicated	The TSTool legend provides more information than simple graphs in Excel or other tools. The user can configure simpler legends with graph properties.	Evaluate whether default legend should be simpler, and allow user to specify a global default.
TSTool does not draw data in middle of interval (e.g., should place point in middle of month, not at start)	TSTool graphing tools can display large numbers of data points and are intended more for scientific visualization than business data visualization.	Implement a graph option for whether to plot data points at the start, middle, or end of the data interval. This has been done to control bar positions but has not been implemented for point or line graphs.
TSTool does not by default format numbers as MM/DD/YYYY	The MM/DD/YYYY format is typical for dates in the USA. However, this format does not sort well and the ISO standard YYYY-MM-DD format is used in TSTool to promote consistency.	Provide optional graph properties to override default date/time format.
TSTool does not by default format numbers with commas to separate thousands.	This was not implemented due to resource limitations and because the default of no separator is appropriate in any locale.	Provide optional properties to display separators.
TSTool does not offer pie charts and some other Excel charts	TSTool was initially developed for time series visualization. More recent enhancements have enabled features to process tables and statistics and such information may be desirable to display in less data-intensive graphs such as pie charts.	Add pie charts and other graph types as part of table visualization enhancements.