10 Excel Integration

Version 10.15.00, 2013-01-10

TSTool and Microsoft Excel offer 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 Excel and describes how they can be integrated.

Many people and software solutions have similar needs when integrating TSTool with Excel:

- 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 are either binary (especially older versions) or complex XML (newer versions), often without clear documentation – it is necessary to reverse engineer Microsoft formats
- The complexity of Excel files make it difficult to simply manipulate the files – intermediate software is needed

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) 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.
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

Feature	TSTool	Excel
		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
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., `NaN`, `-999`).	No specific handling, typically must ensure blanks in data cells so that missing values will not generate errors.
Data units	Handled transparently in most cases,	No specific handling. User must

Feature	TSTool	Excel
	based on units in time series metadata, with checks in place to prevent incompatible manipulation.	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).

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), for example:

- 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.
        - Does anyone have a good reference for EXCEL SQL statements, in particular showing complex queries and how to write to the sheet?
    - Optionally use the `TableToTimeSeries()` command to convert the table to time series.
- Python xlrd (http://pypi.python.org/pypi/xlrd) and xlwt (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
- Apache POI (see: http://poi.apache.org/):
    - Java, so can be directly integrated into TSTool
    - Currently being integrated with TSTool in Excel read/write commands
- Internal TSTool features to write simple Excel files, similar to "Export to Excel" features in many software tools:
    - Features may be implemented in TSTool; however, because TSTool focuses on time series processing, additional integration and flexibility are likely needed than for a simple Export format data dump.
    - Time series and tables can be written to CSV files, which can be imported into Excel.

For the above, 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.

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 CSV 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 and command files are documented. Excel macros can be written to create these files and then TSTool can be called from Excel as described above.

Does anyone have TSTool macros and accompanying documentation and examples that should be distributed with TSTool?

This page is intentionally blank.