SpreadServe Documentation

Release 0.1.0

John O'Sullivan

Aug 10, 2017

Contents

Contents:

CHAPTER 1

Installing The SpreadServe Addin

Installing the addin for the first time

- Get the XLL from http://spreadserve.com/s3/downloads.html or source from https://github.com/SpreadServe/SSAddin
- Install SSAddin.xll as an Excel addin.
  - Use SSAddin64.xll if you're running a 64 bit Excel.
  - Watch this video if you're unsure about adding an addin https://www.youtube.com/watch?v=i_sijj1NZFM
  - Put SSAddin.xll.config in the same directory as SSAddin.xll, and edit it to add Tiingo, Quandl and Barametrics keys if you use those services.
- Create a new sheet, or load one of the test sheets to check that the addin is loaded.
  - Hit *fx* on the formula bar to get the Insert Function dialog.
  - Select the SpreadServe Addin function category.
  - You should see *s2cron*, *s2quandl* and other SpreadServe Addin functions listed.
- Bear in mind that the SpreadServe Addin does not add a ribbon menu. It's designed to work entirely through worksheet functions.

SpreadServe Addin test sheets

There are several test spreadsheets in the zip in the xls directory. These sheets all use RTD updates, so make sure you are in automatic calculation mode. Go to `Formulas/Calculation Options` in Excel and select Automatic. Use ctrl-alt-F9 to recalc everything and force the RTD subscriptions through.

- `cron1.xls`: demonstrates the use of the `s2cron` and `s2sub` functions to set up and track a timer that goes off every 20 seconds. The timer will stop at the end of the day.
- `cron2.xls`: uses of the `s2cron` and `s2sub` functions to set up and track a timer that goes off every 5 seconds. Note how the start and end dates are set in the s2cfg sheet so the timer will run beyond the end of the day, for as many days as the sheet is running.
- `cron3.xls`: uses of the `s2cron` and `s2sub` functions to set up and track a timer that goes off daily at 1430. Note how the start and end dates are set in the s2cfg sheet.

- `quandl1.xls`: uses `s2quandl` to launch a quandl query on a background thread in the subs sheet, and `s2cache` to pull the query result set into cells on the data sheet. You may have to ctrl-alt-F9 a second time to force `s2qcache` execution in the subs sheet.
- `quandl2.xls`: a variation on quandl1. The two differences are the offsetting of the result set in the `data` sheet, and the use of `s2vqcache` instead of `s2qcache`. The offsetting allows result sets to appear anywhere in a sheet instead of being anchored to the top left cell. `s2vqcache` is a volatile version of `s2qcache`. Use of the volatile function avoids the need for a second ctrl-alt-F9.
- `quandl3.xls`: combines the cron and quandl features to implement a quandl query that is executed every 30 seconds.
- `wsock.xls`: uses the *s2websock* function to subscribe to updates from an automated sheet hosted by SpreadServe.
- `tiingows1.xls`: uses the *s2twebsock* and *s2sub* functions to subscribe to live ticking IEX market data from Tiingo. NB you will need to put your Tiingo authorization token into the s2cfg sheet to connect to Tiingo, and you'll need to be permissioned for IEX data at Tiingo.
- `tiingows_option1.cls`: using *s2twebsock* and *s2sub* to drive a Black Scholes option calc with ticking IEX market data.

Some of the example sheets have `_proxy` suffixed to the name. These alternate versions are designed to work from behind an internet proxy. They have extra config sheet entries to configure username, password and proxy connection details. If you're in a corporate environment you'll probably need to use these.

CHAPTER 2

SpreadServe Addin Worksheet Functions

These are the functions you can invoke directly from cells in your spreadsheet.

s2about: get version information.

Parameters

- None

Return value: a string detailing the SpreadServe Addin version, and the version of Excel hosting the adding.

s2cron: setup scheduled timer.

Parameters

- `CronKey`: a value or cell reference evaluating to a string that matches a value in column C of the s2cfg sheet. The s2cfg row with the matching column C value will be used to specify a cron job. See the cron1, 2 or 3 example sheets.

Return value: "OK" if the function succeeds, an Excel error otherwise.

s2quandl: launch a quandl query.

Parameters

- `QueryKey`: a value or cell reference evaluating to a string that matches a value in column C of the s2cfg sheet. The s2cfg row with the matching column C value will be used to specify a quandl query. See the quandl1, 2 or 3 example sheets.
- `Trigger`: an optional trigger. The value isn't used inside the function, but a change in the input can be used to force repeat execution. See the quandl3.xls sheet for an example of an s2quandl trigger parameter hooked up to s2cron output to rerun a query on a timed basis.

Return value: "OK" if the function succeeds, an Excel error otherwise.

s2qcache: get a value from a quandl query result set. The position of the cell invoking this function is used to figure out which cell to get from the result set.

Parameters

- `QueryKey`: should match the QueryKey given to *s2quandl*.

- `XOffset`: defaults to 0. If the left hand side of the result grid on your sheet is not column A this should be the number of columns across.
- `YOffset`: defaults to 0. If the top row of the result grid on your sheet is not row 1 this should be the number of rows down.
- `Trigger`: an optional trigger. The value isn't used inside the function, but a change in the input can be used to force repeat execution. See the quandl3.xls sheet for an example of an s2quandl trigger parameter hooked up to s2cron output to rerun a query on a timed basis.

Return value: a value from the result set, or #N/A.

s2vqcache: a volatile version of s2qcache.

Parameters

- `QueryKey`: should match the QueryKey given to *s2quandl*.
- `XOffset`: defaults to 0. If the left hand side of the result grid on your sheet is not column A this should be the number of columns across.
- `YOffset`: defaults to 0. If the top row of the result grid on your sheet is not row 1 this should be the number of rows down.

Return value: a value from the result set, or #N/A.

s2tiingo: launch a tiingo query.

Parameters

- `QueryKey`: a value or cell reference evaluating to a string that matches a value in column C of the s2cfg sheet. The s2cfg row with the matching column C value will be used to specify a tiingo query. See the tiingo1 or 2 example sheets.
- `Trigger`: an optional trigger. The value isn't used inside the function, but a change in the input can be used to force repeat execution.

Return value: "OK" if the function succeeds, an Excel error otherwise.

s2tcache: get a value from a tiingo query result set. The position of the cell invoking this function is used to figure out which cell to get from the result set.

Parameters

- `QueryKey`: should match the QueryKey given to *s2tiingo*.
- `XOffset`: defaults to 0. If the left hand side of the result grid on your sheet is not column A this should be the number of columns across.
- `YOffset`: defaults to 0. If the top row of the result grid on your sheet is not row 1 this should be the number of rows down.
- `Trigger`: an optional trigger. The value isn't used inside the function, but a change in the input can be used to force repeat execution.

Return value: a value from the result set, or #N/A.

s2vtcache: a volatile version of s2tcache.

Parameters

- `QueryKey`: should match the QueryKey given to *s2tiingo*.
- `XOffset`: defaults to 0. If the left hand side of the result grid on your sheet is not column A this should be the number of columns across.

- `YOffset`: defaults to 0. If the top row of the result grid on your sheet is not row 1 this should be the number of rows down.

Return value: a value from the result set, or #N/A.

s2baremetrics: launch a Baremetrics metric query.

Parameters

- `QueryKey`: a value or cell reference evaluating to a string that matches a value in column C of the s2cfg sheet. The s2cfg row with the matching column C value will be used to specify a Baremetrics query. See the baremetrics_summary1 or baremetrics_metric1 example sheets.
- `Trigger`: an optional trigger. The value isn't used inside the function, but a change in the input can be used to force repeat execution.

Return value: "OK" if the function succeeds, an Excel error otherwise.

s2bcache: get a value from a Baremetrics query result set.

Parameters

- `QueryKey`: should match the QueryKey given to **s2baremetrics**.
- `Date`: Baremetrics result sets are keyed on date; think of date as picking out a row. You should supply a string in `yyyy-MM-dd` format, or use the `s2today` function. Don't use Excel's volatile `TODAY` function as you'll cause an endless recalc cycle.
- `Field`: pick out a column in the result set row selected by `Date`.
- `Trigger`: an optional trigger. The value isn't used inside the function, but a change in the input can be used to force repeat execution.

Return value: a value from the result set, or #N/A.

s2vbcache: a volatile version of **s2bcache**.

Parameters

- `QueryKey`: should match the QueryKey given to **s2baremetrics**.
- `Date`: Baremetrics result sets are keyed on date; think of date as picking out a row. You should supply a string in `yyyy-MM-dd` format, or use the **s2today** function. Don't use Excel's volatile `TODAY` function as you'll cause an endless recalc cycle.
- `Field`: pick out a column in the result set row selected by `Date`.

Return value: a value from the result set, or #N/A.

s2sub: subscribe to RTD updates generated by s2cron, s2quandl or s2websock.

Parameters

- `SubCache`: [quandl|cron|websock]
- `CacheKey`: should match the CronKey or QueryKey given to s2cron or s2quandl.
- `Property`: [status|count|next|last|mX_Y_Z] count: cron event count for s2cron, rows in result set for s2quandl. next: time of next cron event. last: time of last cron event.

Return value: RTD value, or #N/A.

s2websock: subscribe via WebSockets to a page in a SpreadServe hosted sheet.

Parameters

- `SockKey`: a value or cell reference evaluating to a string that matches a value in column C of the s2cfg sheet. The s2cfg row with the matching column C value will be used to specify the URL of a page in a SpreadServe hosted spreadsheet. See the websock1 example sheet.

Return value: "OK" if the function succeeds, an Excel error otherwise.

s2twebsock: subscribe via WebSockets to a Tiingo market data feed.

Parameters

- `SockKey`: a value or cell reference evaluating to a string that matches a value in column C of the s2cfg sheet. The s2cfg row with the matching column C value will be used to specify the URL for the Tiingo websocket connection. See the tiingows1 example sheet.

Return value: "OK" if the function succeeds, an Excel error otherwise.

s2wscache: get a value from a WebSocket subscription cache.

Parameters

- `SockKey`: should match the SockKey given to *s2websocket*.
- `CellKey`: for instance, m2_6_0 for col 3, row 7 on first sheet. Use 'Page Source' in your browser to examine the HTML on a page you want to subscribe to, and look for the div id tags to figure out the value you need.
- `Trigger`: an optional trigger.

Return value: a value from the cache, or #N/A.

s2vwscache: a volatile version of `s2wscache`.

Parameters

- `SockKey`: should match the SockKey given to *s2websocket*.
- `CellKey`: for instance, m2_6_0 for col 3, row 7 on first sheet. Use 'Page Source' in your browser to examine the HTML on a page you want to subscribe to, and look for the div id tags to figure out the value you need.

Return value: a value from the cache, or #N/A.

s2today: non volatile alternative to Excel's *TODAY*.

Parameters

- `Offset`: 0 to get today, -1 for yesterday, +1 for tomorrow, -7 for a week ago, +7 for a week from now.

Return value: a yyyy-MM-dd formatted date string.

CHAPTER 3

SpreadServe Addin Configuration

Log files

The SSAddin creates log files in your `%TEMP%` directory. To find them do this in a DOS box:

```
echo %TEMP%
cd %TEMP%
dir ssaddin* /od
dir *.csv
```

Note that the process ID of the Excel instance hosting your SSAddin is embedded in the log file name. The log file captures all the RTD updates sent by the addin to the sheet, together with their values. It also logs the start and end of Quandl queries. The addin also dumps the result sets returned from Quandl into CSV files in the `%TEMP%` directory. The files are named `<QueryKey>_<ProcessID>.csv`.

s2cfg sheet

Any spreadsheet that uses SSAddin must have a sheet called `s2cfg`. The SSAddin worksheet functions get their configuration from the s2cfg sheet and will fail if it doesn't exist or if its contents are not correctly laid out. The log files should alert you if there's a problem in your s2cfg sheet. They are also a good way of checking that the addin has composed your quandl or tiingo queries as you expected. Bear in mind these points on how the addin scans the s2cfg sheet for configuration. Also check the example sheets in the xls sub directory for concrete illustrations of the guidelines below.

- SSAddin scans the s2cfg sheet from the first row downwards. It will stop scanning when it finds a row with an empty cell in column A. This means you can't have spaces between your config. It must all be in a single contiguous block from row 1 downwards.
- The value in column A must be `quandl`, `tiingo`, `cron`, `websock` or `twebsock`.
- Depending on the value in column A there are different expectations for the values in column B onwards.
  - `quandl`: column B should be `query` or `config`
    - `query`: column C should be the unique QueryKey that's passed to the `s2quandl` function, column D should be `dataset` and column E should name a Quandl dataset eg `FRED/DED1` or `OPEC/ORB`. Any further columns should give key value pairs to tacked on to the Quandl query URL after the `?`

For instance column F could be `rows` and column G `5` so that `?rows=5` is appended to the URL query submitted to quandl.

  * `config`: column pairs from C & D onwards are reserved for name value pairs that apply to all queries. Currently only `auth_token` is supported. If you put `auth_token` in column C, then put your actual key in column D for it to be added to all queries. However, we recommend you put your key in SSAddin.xll.config instead, so you don't indavertently share your key when sharing your spreadsheet.

- `tiingo`: column B should be `query` or `config`
  * `query`: column C should be the unique QueryKey that's passed to the `s2tiingo` function, column D should be `ticker` and column E should be a ticker symbol eg `msft` or `aapl`. The ticker symbol should be lower case. Column F should be `root`, followed by `daily` or `funds` in column G. Column H is optional. If it's present it should be `leaf` and then column I should be `prices`. If it's absent a tiingo query that gets meta data for the symbol will be dispatched. Finally, columns J, K, L & M can be used to specify startDate and endDate for historical price queries.
  * `config`: column pairs from C & D onwards are reserved for name value pairs that apply to all queries or Tiingo web socket connections (see twebsock below). Supported config keys are...
    - `auth_token`: put `auth_token` in column C, and your actual key in column D for it to be added to all queries or used by twebsock.
    - `http_proxy_host`: if this appears in column C then column D should give a proxy hostname. SSAddin will then connect via the proxy rather than direct to the internet.
    - `http_proxy_port`: port for the proxy connection.
    - `http_proxy_user`: user name for the proxy connection. Often this is in DOMAINUSER format for Windows Active Directory user IDs.
    - `http_proxy_password`: password for the proxy connection.
- `baremetrics`: column B should be `query` or `config`
  * `query`: column C should be the unique QueryKey that's passed to the `s2baremetrics` function, column D should be `qtype` and column E should be a `summary`, `plan` or `metric`. For a qtype of `plan` or `metric` you need a following key/value pair that specifies which metric. The key, in column F should be `metric`, and then in column G you should specify `mrr`, *arpu'*, `ltv` or any of the available metrics. Columns J, K, L & M can be used to specify `start_date` and `end_date` with the date values in columns K and M supplied by `s2today` or handcoded yyy-MM-dd strings. Don't use Excel's own *TODAY* function for these as it's volatile and will cause an endless calc cycle. Finally, if you're testing against the Sandbox API put `sandbox` in column L and `TRUE` in column M and ensure you have your Sandbox API key in SSAddin.xll.config. Don't forget to remove `sandbox:TRUE` and switch the API key in SSAddin.xll.config when you've finished testing!
  * `config`: column pairs from C & D onwards are reserved HTTP proxy settings. See details for `tiingo` above.
- `twebsock`: when column B contains `tiingo` then column C specifies a `SockKey` to pass to `s2twebsock`. Column D should give the URL for the Tiingo API socket eg `wss://api.tiingo.com/iex`
- `cron`: when column B contains `tab` then column C should have a unique `CronKey` that will be passed to the `s2cron` worksheet function which will then get the cron job specification from columns D to K. This job spec is then passed to SSAddin's internal NCrontab implementation. Bear in mind that SSAddin uses a hacked version of NCrontab that extends the spec to add seconds.
  * `D`: seconds
  * `E`: minutes

    * `F`: hours
    * `G`: days
    * `H`: month
    * `I`: weekday
    * `J`: start - defaults to the start of today, today being the day when the process started.
    * `K`: end - defaults to the end of today
  - `websock`: when column B contains `url` then column C specifies a `SockKey` to pass to `s2websock`. Column D should give the hostname of a SpreadServe server, column E the port number, and column F the rest of the URL, often referred to as the path.

Note that if column B has any other value than described above it will be ignored. One convention you'll see in the SSAddin example s2cfg sheets is `comment` occurring in column B so that the rest of the row can be used as headers to describe the real values below.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search